Algorithm 902: GPOPS, A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using the Gauss Pseudospectral Method

ANIL V. RAO
University of Florida
DAVID A. BENSON
The Charles Stark Draper Laboratory, Inc.
CHRISTOPHER DARBY, MICHAEL A. PATTERSON,
CAMILA FRANCOLIN, and ILYSSA SANDERS
University of Florida
and
GEOFFREY T. HUNTINGTON
Blue Origin, LLC

An algorithm is described to solve multiple-phase optimal control problems using a recently developed numerical method called the *Gauss pseudospectral method*. The algorithm is well suited for use in modern vectorized programming languages such as FORTRAN 95 and MATLAB. The algorithm discretizes the cost functional and the differential-algebraic equations in each phase of the optimal control problem. The phases are then connected using linkage conditions on the state and time. A large-scale nonlinear programming problem (NLP) arises from the discretization and the significant features of the NLP are described in detail. A particular reusable MATLAB implementation of the algorithm, called *GPOPS*, is applied to three classical optimal control problems to demonstrate its utility. The algorithm described in this article will provide researchers and engineers a useful software tool and a reference when it is desired to implement the Gauss pseudospectral method in other programming languages.

The authors gratefully acknowledge support for this research from the Army Research Office under Grant 55173-CI and The U.S. Office of Naval Research under Grant N00014-08-1-1173.
Authors' addresses: A. V. Rao, C. Darby, M.A. Patterson, C. Francolin, and I. Sanders, Department of Mechanical and Aerospace Engineering, University of Florida, P.O. Box 116250, Gainesville, FL 32611-6250; email: {anilvrao, cdarby, mpatterson, francoli}@ufl.edu; ilyssa327@gmail.com; D. A. Benson, The Charles Stark Draper Laboratory, Inc., 555 Technology Square, Cambridge, MA 02139-3563; email: dbenson@draper.com; G. T. Huntington, Blue Origin, LLC, 21218 76th Ave. S., Kent, Washington 98032-2442; email: ghuntington@blueorigin.com.

DOI 10.1145/1731022.1731032 http://doi.acm.org/10.1145/1731022.1731032

Categories and Subject Descriptors: G.1.4 [**Numerical Analysis**]: Quadrature and Numerical Differentiation; G.1.6 [**Numerical Analysis**]: Optimization—*Nonlinear programming*

General Terms: Algorithms, Languages

Additional Key Words and Phrases: Dynamic optimization, nonlinear optimization, optimal control, nonlinear programming, phases, computational methods

ACM Reference Format:

Rao, A. V., Benson, D. A., Darby, C., Patterson, M. A., Francolin, C., Sanders, I., and Huntinton, G. T. 2010. Algorithm 902: GPOPS, a MATLAB software for solving multiple-phase optimal control problems using the Gauss pseudospectral method. ACM Trans. Math. Softw. 37, 2, Article 22 (April 2010), 39 pages.

DOI = 10.1145/1731022.1731032 http://doi.acm.org/10.1145/1731022.1731032

1. INTRODUCTION

Due to the increasing complexity of engineering applications, over the past two decades the subject of optimal control has transitioned from theory to computation. In particular, computational optimal control has become a science in and of itself, resulting in a variety of numerical methods and corresponding software implementations of these methods. The vast majority of software implementations of optimal control today are those that involve the direct transcription of a continuous-time optimal control problem to a nonlinear program (NLP). The NLP is then solved using one of a variety of well-known software packages [Gill et al. 2002; Byrd et al. 2006; Betts and Frank 1994].

Over the last decade, a particular class of direct collocation methods, called *pseudospectral methods* [Canuto et al. 1988, 2007; Fornberg 1998; Trefethen 2001], have risen to prominence in the numerical solution of optimal control problems [Elnagar et al. 1995; Elnagar and Kazemi 1998; Vlassenbroeck and Doreen 1988; Fahroo and Ross 2000, 2001; Ross and Fahroo 2004b; Rao 2003; Williams 2004a, 2004b, 2005; Ross and Fahroo 2004a, 2004b; Ross and Fahroo 2008a, 2008b; Benson 2004; Benson et al. 2006; Huntington 2007; Kameswaran and Biegler 2008]. In a pseudospectral method, the state and control are approximated using global polynomials and collocation of the differential-algebraic equations is performed at *orthogonal collocation points* (i.e., the collocation points are the roots of an orthogonal polynomial and/or a linear combination of an orthogonal polynomial and its derivatives). Pseudospectral methods, that is, the *combination* of using global polynomials with orthogonally collocated points, are known to converge *spectrally* (i.e., converging to the solution faster than any power of N^{-m} where N is the number of collocation points and m is any finite value [Fornberg 1994]). Furthermore, pseudospectral methods have been shown to be mathematically sound [Fornberg 1994, 1998; Trefethen 2001] and have been used extensively in fluid dynamics [Canuto et al. 1988, 2007; Don 2000]. In situations where global collocation is inadequate (e.g., problems where the solution has steep gradients or possible discontinuities), pseudospectral methods are employed in the form of multidomain techniques (i.e., the problem is divided into a relatively small number of subintervals and global collocation is performed across each subinterval) [Canuto et al. 2007].[1] Finally,

[1]The multidomain approach differs significantly from the *local collocation* approach [Betts 2001]

it is noted that, in the context of optimal control problems, a study has been performed that supports the use of global (semi-global) spectral collocation over local spectral collocation [Huntington and Rao 2008a].

The three most commonly used set of orthogonal collocation points in a pseudospectral method are *Legendre-Gauss* (LG), *Legendre-Gauss-Radau* (LGR), and *Legendre-Gauss-Lobatto* (LGL) points. These three sets of points are obtained from the roots of a Legendre polynomial and/or linear combinations of a Legendre polynomial and its derivatives. All three sets of points are defined on the domain $[-1, 1]$, but differ significantly in that the LG points include *neither* of the endpoints, the LGR points include *one* of the endpoints, and the LGL points include *both* of the endpoints. In addition, the LGR points are asymmetric relative to the origin and are not unique in that they can be defined using either the initial point or the terminal point. As a result of these three collocation points, the following three mathematical methods have been developed: the *Legendre pseudospectral method* (LPM) [Elnagar et al. 1995; Fahroo and Ross 2001], the *Radau pseudospectral method* (RPM) [Kameswaran and Biegler 2008], and the *Gauss pseudospectral method* (GPM) [Benson 2004; Benson et al. 2006; Huntington 2007]. In addition, methods have been developed to handle interior point constraints [Fahroo and Ross 2000; Ross and Fahroo 2004a], and nonsequential phases [Rao 2003].

While many pseudospectral methods and subsequent extensions have been described mathematically, these methods have to date not been described in the open literature in the form of *algorithms* that enable implementation in mathematical software. As a result, it is difficult for a researcher or an engineer to study these mathematical methods and arrive at a tractable software implementation after any reasonable investment of time. Instead, most users typically resort to using either commercial off-the-shelf (COTS) or open-source software to solve optimal control problems. Examples of COTS software include *SOCS* [Betts and Huffman 1997], *DIRCOL* [von Stryk 2000], *GESOP* [Jansch et al. 1994], *OTIS* [Vlases et al. 1990], *DIDO* [Ross and Fahroo 2001], *DIRECT* [Williams 2008], and *MISER* [Goh and Teo 1988], while examples of open-source optimal control software include *DYNOPT* [Cizniar et al. 2006], *OPTCONTROLCENTRE* [Jockenhovel 2002], and *PSOPT* [Becerra 2009]. The typical alternative to COTS or open-source software is "home-grown" software.

While users benefit greatly from COTS software, such programs require a great investment of time to learn. Moreover, even after overcoming the steep learning curve associated with *using* these programs, a user has gained little insight into the underlying algorithm used in the software. Thus the underlying algorithm remains a "black box" and the user would still find it difficult to implement such an algorithm independently. Even worse, because using a "black-box" provides little insight about the underlying methodology, COTS programs are limited in their educational value.

---

in that in local collocation the problem is divided into many small subintervals whereas in the multidomain technique the problem is divided into few relatively large subdomains.

The purpose of this article is to enhance the understanding of the implementation of pseudospectral methods in mathematical software for solving optimal control problems. To achieve this goal, this article provides a detailed description of an algorithm and a supporting software implementation of the algorithm. In particular, the algorithm developed in this article utilizes the aforementioned *Gauss pseudospectral method* (GPM) [Benson 2004; Benson et al. 2006; Huntington 2007] for solving multiple-phase optimal control problems. The GPM was chosen as the basis for the algorithm of this article because it has many useful numerical properties including the ability to generate highly accurate costates (adjoints) [Benson 2004; Benson et al. 2006; Huntington 2007] of the continuous-time optimal control problem.[2] It is noted that the authors of this article have already developed customized (i.e., nonreusable) implementations of the algorithm described in this article for applications in flight dynamics [Huntington 2007; Huntington et al. 2007; Huntington and Rao 2007, 2008b], but until now no detailed information has been provided to the research community on how to implement the GPM in a general-purpose software program. As a result, this article is an attempt to fill the gap between the theory and the implementation of pseudospectral methods for optimal control by providing a general-purpose vectorized implementation of the Gauss pseudospectral method [Benson 2004; Benson et al. 2006; Huntington 2007; Huntington et al. 2007; Huntington and Rao 2007, 2008b]. The algorithm described in this article can be used in modern vectorized programming languages such as FORTRAN 95/98/2000 or MATLAB. A particular MATLAB implementation, called *GPOPS*, is provided and is found to work well on a variety of complex multiple-phase continuous-time optimal control problems. After describing the algorithm, three examples are provided to demonstrate the flexibility and utility of the software. The first example is a modified version of the chemical engineering fed-batch reactor problem known as the *Lee-Ramirez Bioreactor* [Balsa-Canto et al. 2001]. The second and third examples are aerospace engineering problems of a multiple-stage launch vehicle ascent [Benson 2004; Huntington 2007; Huntington and Rao 2007] and a classical one-dimensional sounding rocket ascent (known as the *Goddard rocket problem* [Betts 2001]).[3] In order to see the power of the software, the first example is compared against the commercially available software *PROPT* [Rutquist and Edvall 2008], while the second and third examples are compared against the well-known and commercially available program *Sparse Optimal Control Software* SOCS [Betts and Huffman 1997]. It is found that the solutions obtained using GPOPS compare well with those obtained using *PROPT* and SOCS, respectively. The results of this article demonstrate that GPOPS is useful for solving optimal control problems that arise in different branches of engineering.

[2]With some modifications, the algorithm described in this article can be adapted to other pseudospectral methods such as the aforementioned RPM [Kameswaran and Biegler 2008] or the LPM [Elnagar et al. 1995; Fahroo and Ross 2001].

[3]It is well known that the Goddard rocket problem has a segment that consists of a singular arc and this example is a demonstration of the ability of GPOPS to handle problems with singular arcs.

2. GENERAL MULTIPLE-PHASE OPTIMAL CONTROL PROBLEMS

The problems that GPOPS is capable of solving fall into the following general category. Given a set of P phases (where $p \in [1, \ldots, P]$), minimize the cost functional

$$J = \sum_{p=1}^{P} \left[\Phi^{(p)}\left(\mathbf{x}^{(p)}(t_0), t_0^{(p)}, \mathbf{x}^{(p)}(t_f), t_f^{(p)}; \mathbf{q}^{(p)}\right) + \int_{t_0^{(p)}}^{t_f^{(p)}} \mathcal{L}^{(p)}\left(\mathbf{x}^{(p)}(t), \mathbf{u}^{(p)}(t), t; \mathbf{q}^{(p)}\right) dt \right] \tag{1}$$

subject to the dynamic constraints

$$\dot{\mathbf{x}}^{(p)} = \mathbf{f}^{(p)}\left(\mathbf{x}^{(p)}, \mathbf{u}^{(p)}, t; \mathbf{q}^{(p)}\right), \qquad (p = 1, \ldots, P), \tag{2}$$

the inequality path constraints

$$\mathbf{C}_{\min}^{(p)} \le \mathbf{C}^{(p)}\left(\mathbf{x}^{(p)}(t), \mathbf{u}^{(p)}(t), t; \mathbf{q}^{(p)}\right) \le \mathbf{C}_{\max}^{(p)}, \qquad (p = 1, \ldots, P), \tag{3}$$

the boundary conditions

$$\phi_{\min}^{(p)} \le \phi^{(p)}\left(\mathbf{x}^{(p)}(t_0), t_0^{(p)}, \mathbf{x}^{(p)}(t_f), t_f^{(p)}; \mathbf{q}^{(p)}\right) \le \phi_{\max}^{(p)}, \qquad (p = 1, \ldots, P), \tag{4}$$

and the linkage constraints [Betts 2001]

$$\mathbf{L}_{\min}^{(s)} \le \mathbf{L}^{(s)}\left(\mathbf{x}^{(p_l^s)}(t_f), t_f^{(p_l^s)}; \mathbf{q}^{(p_l^s)}, \mathbf{x}^{(p_r^s)}(t_0), t_0^{(p_r^s)}; \mathbf{q}^{(p_r^s)}\right) \le \mathbf{L}_{\max}^{(s)}, \begin{cases} p_l, p_r \in [1, \ldots, P], \\ s = 1, \ldots, L. \end{cases} \tag{5}$$

where $\mathbf{x}^{(p)}(t) \in \mathbb{R}^{n_x^{(p)}}$, $\mathbf{u}^{(p)}(t) \in \mathbb{R}^{n_u^{(p)}}$, $\mathbf{q}^{(p)} \in \mathbb{R}^{n_q^{(p)}}$, and $t \in \mathbb{R}$ are, respectively, the state, control, static parameters, and time in phase $p = [1, \ldots, P]$, L is the number of pairs of phases to be linked, and $(p_l^{(s)}, p_r^{(s)}) \in [1, \ldots, P]$, $s = 1, \ldots, L$, are the "left" and "right" phase numbers, respectively. Specifically, the functions $\Phi^{(p)}$, $\mathcal{L}^{(p)}$, $\mathbf{f}^{(p)}$, $\mathbf{C}^{(p)}$, $\phi^{(p)}$, and $\mathbf{L}^{(s)}$ are defined by the following mappings:

$$\begin{aligned}
\Phi^{(p)} &: \mathbb{R}^{n_x^{(p)}} \times \mathbb{R} \times \mathbb{R}^{n_x^{(p)}} \times \mathbb{R} \times \mathbb{R}^{n_q^{(p)}} &&\longrightarrow \mathbb{R}, \\
\mathcal{L}^{(p)} &: \mathbb{R}^{n_x^{(p)}} \times \mathbb{R}^{n_u^{(p)}} \times \mathbb{R} \times \mathbb{R}^{n_q^{(p)}} &&\longrightarrow \mathbb{R}, \\
\mathbf{f}^{(p)} &: \mathbb{R}^{n_x^{(p)}} \times \mathbb{R}^{n_u^{(p)}} \times \mathbb{R} \times \mathbb{R}^{n_q^{(p)}} &&\longrightarrow \mathbb{R}^{n_x^{(p)}}, \\
\mathbf{C}^{(p)} &: \mathbb{R}^{n_x^{(p)}} \times \mathbb{R}^{n_u^{(p)}} \times \mathbb{R} \times \mathbb{R}^{n_q^{(p)}} &&\longrightarrow \mathbb{R}^{n_c^{(p)}}, \\
\phi^{(p)} &: \mathbb{R}^{n_x^{(p)}} \times \mathbb{R} \times \mathbb{R}^{n_x^{(p)}} \times \mathbb{R} \times \mathbb{R}^{n_q^{(p)}} &&\longrightarrow \mathbb{R}^{n_\phi^{(p)}}, \\
\mathbf{L}^{(s)} &: \mathbb{R}^{n_x^{(p_l^s)}} \times \mathbb{R} \times \mathbb{R}^{n_q^{(p_l^s)}} \times \mathbb{R}^{n_x^{(p_r^s)}} \times \mathbb{R} \times \mathbb{R}^{n_q^{(p_r^s)}} &&\longrightarrow \mathbb{R}^{n_L^{(s)}},
\end{aligned} \tag{6}$$

where $n_x^{(p)}$, $n_u^{(p)}$, $n_q^{(p)}$, $n_c^{(p)}$, and $n_\phi^{(p)}$ are the dimensions of the state, control, static parameter vector, path constraint vector, and boundary condition vector in phase $p = [1, \ldots, P]$, and $n_L^{(s)}$ is the dimension of the vector formed by the sth set of linkage constraints. While much of the time a user may want to solve a problem consisting of multiple phases, it is important to note that the phases *need not be sequential*. To the contrary, any two phases may be linked provided that the independent variable does not change direction (i.e., the independent variable moves in the same direction during each phase that is linked). It is

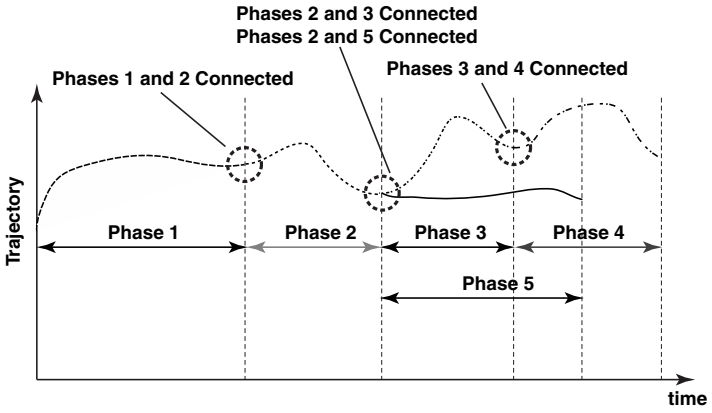

Fig. 1. Schematic of linkages for multiple-phase optimal control problem. The example shown in the picture consists of five phases where the ends of phases 1, 2, and 3 are linked to the starts of phases 2, 3, and 4, respectively, while the end of phase 2 is linked to the start of phase 5.

noted that the approach to linking phases used in GPOPS is based on well-known formulations in the literature such as those given in Betts [2001] and Betts [1998]. A schematic of how phases can potentially be linked is given in Figure 1.

3. GAUSS PSEUDOSPECTRAL DISCRETIZATION

Before proceeding to a description of the GPOPS algorithm, in this section we provide a detailed description of the discretization of a multiple-phase optimal control problem via the GPM. For completeness, we restate some of the basic equations that have been previously developed for the GPM [Benson 2004; Benson et al. 2006; Huntington 2007].

Let $p \in [1, \ldots, P]$ be a particular phase of an optimal control problem and let $(\cdot)^{(p)}$ denote information in the pth phase. For every phase of the problem, consider now the following transformation of the independent variable, t, to the variable $\tau \in [-1, 1]$[4]:

$$t^{(p)} = \frac{t_f^{(p)} - t_0^{(p)}}{2}\tau^{(p)} + \frac{t_f^{(p)} + t_0^{(p)}}{2}. \tag{7}$$

Furthermore, suppose we choose the collocation points in each phase to be the set of *Legendre-Gauss* (LG) points, $(\tau_1, \ldots, \tau_{N^{(p)}})$, which are the roots of the Nth-degree Legendre polynomial, $P_N(\tau)$, given as[5]

$$P_N = \frac{1}{2^N N!}\frac{d^N}{d\tau^N}\left\{\left[\tau^2 - 1\right]^N\right\}. \tag{8}$$

[4]For simplicity with the remainder of this section, superscript "(p)" denoting the phase number will be suppressed unless it is needed to distinguish between phases.

[5]The Legendre-Gauss (LG) points are conveniently obtained by computing the eigenvalues of the $N \times N$ *Jacobi matrix* as is done in the pseudospectral differentiation matrix suite [Weideman and Reddy 2000].

Corresponding to the LG points are the LG *weights* which are computed as

$$w_i = \frac{2}{\left(1-\tau_i^2\right)\left[P_N'\right]^2} \qquad (i = 1, \ldots, N). \tag{9}$$

Finally, the *discretization points* used in the GPM are the LG points plus the points $\tau_0 = -1$ and $\tau_{N+1} = 1$ (thus resulting in a complete set of points $(\tau_0, \ldots, \tau_{N+1})$.

The discretization in each phase $p = [1, \ldots, P]$ of the problem can then be stated in terms of the independent variable τ as follows. First, the state is approximated using a basis of $N+1$ Lagrange interpolating polynomials, $L_i(\tau)\,(i = 0, \ldots, N)$,

$$\mathbf{x}(\tau) \approx \mathbf{X}(\tau) = \sum_{i=0}^{N} \mathbf{X}(\tau_i) L_i(\tau), \tag{10}$$

where $L_i(\tau)\,(i = 0, \ldots, N)$ are defined as

$$L_i(\tau) = \prod_{j=0, j\neq i}^{N} \frac{\tau - \tau_j}{\tau_i - \tau_j}. \tag{11}$$

It is known that the Lagrange polynomials $L_i(\tau)\ (i = 0, \ldots, N)$ satisfy the so called *isolation property*

$$L_i(\tau_j) = \begin{cases} 1, & i = j, \\ 0, & i \neq j. \end{cases} \tag{12}$$

The continuous cost functional of Equation (1) is then approximated using the values of the state, control, and time at the LG points via a Gauss quadrature [Davis and Rabinowitz 1984] as

$$\begin{aligned} J \;=\; & \sum_{p=1}^{P} \Phi^{(p)}\left(\mathbf{X}_0^{(p)}, t_0^{(p)}, \mathbf{X}_f^{(p)}, t_f^{(p)}\right) \\ & + \sum_{p=1}^{P} \frac{t_f^{(p)} - t_0^{(p)}}{2} \sum_{k=1}^{N^{(p)}} w_k^{(p)} \mathcal{L}^{(p)}\left(\mathbf{X}_k^{(p)},\ \mathbf{U}_k^{(p)}, \tau_k^{(p)}; \mathbf{q}^{(p)}, t_0^{(p)}, t_f^{(p)}\right). \end{aligned} \tag{13}$$

Next, differentiating the expression in Equation (10) with respect to τ gives

$$\frac{d\mathbf{X}}{d\tau} \approx \sum_{i=0}^{N} \mathbf{X}(\tau_i) \frac{dL_i(\tau)}{d\tau}. \tag{14}$$

The derivative of each Lagrange polynomial at the LG points can be represented compactly in the form of a *differentiation matrix*, $\mathbf{D} \in \mathbb{R}^{N \times N+1}$ as

$$\mathbf{D}_{ki} = \dot{L}_i(\tau_k) = \sum_{l=0}^{N} \frac{\displaystyle\prod_{j=0, j\neq i,l}^{N} (\tau_k - \tau_j)}{\displaystyle\prod_{j=0, j\neq i}^{N} (\tau_i - \tau_j)}, \tag{15}$$

where $k = 1, \ldots, N$ and $i = 0, \ldots, N$. The dynamic constraint of Equation (2) is then transcribed into algebraic constraints in each phase $p = [1, \ldots, P]$ of the problem as

$$\sum_{i=0}^{N} \mathbf{D}_{ki}\mathbf{X}_i - \frac{t_f - t_0}{2}\,\mathbf{f}(\mathbf{X}_k,\ \mathbf{U}_k, \tau_k; \mathbf{q}, t_0, t_f) = \mathbf{0} \quad (k = 1, \ldots, N), \tag{16}$$

where $\mathbf{X}_k \equiv \mathbf{X}(\tau_k) \in \mathbb{R}^n$ and $\mathbf{U}_k \equiv \mathbf{U}(\tau_k) \in \mathbb{R}^m$ $(k = 1, \ldots, N)$. Next, define an additional variable $\mathbf{X}_f \equiv \mathbf{X}_{N+1} \equiv \mathbf{X}(\tau_f)$ as

$$\mathbf{X}_0 = \mathbf{X}(\tau_0), \tag{17}$$

$$\mathbf{X}_{N+1} = \mathbf{X}_0 + \frac{t_f - t_0}{2}\sum_{k=1}^{N} w_k\,\mathbf{f}(\mathbf{X}_k,\ \mathbf{U}_k, \tau_k; \mathbf{q}, t_0, t_f). \tag{18}$$

It is noted that, because we have introduced an additional variable, Equation (18) is an additional constraint in the discretization (in order to maintain the same number of degrees of freedom). Now, because Equation (18) is a function of the right-hand side of the differential equation at each LG point, Equation (16) can be used to solve for $\mathbf{f}$ and the result can be substituted into Equation (18). The result of this substitution transforms Equation (18) into a linear equation given as

$$\mathbf{X}_{N+1} - \mathbf{X}_0 - \sum_{i=0}^{N}\sum_{k=1}^{N} w_k\mathbf{D}_{ki}\mathbf{X}_i = \mathbf{0}. \tag{19}$$

Because it is linear, Equation (19) is implemented [as opposed to Equation (18)]. Similarly, the path constraints of Equation (3) are discretized at the LG points as

$$\mathbf{C}_{\min} \le \mathbf{C}(\mathbf{X}_k,\ \mathbf{U}_k, \tau_k; t_0, \mathbf{q}, t_f) \le \mathbf{C}_{\max} \quad (k = 1, \ldots, N). \tag{20}$$

Furthermore, the boundary conditions of Equation (4) are expressed as

$$\phi_{\min} \le \phi(\mathbf{X}_0, t_0, \mathbf{X}_{N+1}, t_f) \le \phi_{\max}. \tag{21}$$

Finally, the linkage constraints are mapped using the values at the termini and start, respectively, of the phase pairs $(p_l, p_r) \in [1, \ldots, P]$ $(l, r = [1, \ldots, P])$ as

$$\mathbf{L}_{\min}^{(s)} \le \mathbf{L}^{(s)}\left(\mathbf{X}_{N+1}^{(p_l^s)}, t_f^{(p_l^s)}; \mathbf{q}^{(p_l^s)}, \mathbf{X}_0^{(p_u^s)}, t_0^{(p_u^s)}; \mathbf{q}^{(p_u^s)}\right) \le \mathbf{L}_{\max}^{(s)}, \quad \begin{cases} p_l, p_u \in [1, \ldots, P], \\ s = 1, \ldots, L. \end{cases} \tag{22}$$

4. ALGORITHM FOR SOLVING MULTIPLE-PHASE OPTIMAL CONTROL PROBLEMS USING THE GAUSS PSEUDOSPECTRAL METHOD

In this section we provide the algorithm for mapping of the multiple-phase GPM to an NLP in standard form. As mentioned in the Introduction, we note that the approach described in this section is a generalization of many custom (nonreusable) MATLAB programs that have been developed by the authors or MATLAB programs codeveloped by the authors and their colleagues [Rao 2003; Benson 2004; Huntington 2007; Huntington et al. 2007; Huntington and Rao

2007, 2008a, 2008b]. It is noted that the algorithm described is a formalization of an extensive amount of programming experience.

In general, an NLP has the following standard form. Minimize the cost function

$$f(\mathbf{Z}) \tag{23}$$

subject to the algebraic equality and inequality constraints

$$\begin{array}{rcrcl} \mathbf{Z}_{\min} & \leq & \mathbf{Z} & \leq & \mathbf{Z}_{\max}, \\ & & \mathbf{c}_E(\mathbf{Z}) & = & \mathbf{0}, \\ \mathbf{c}_{I,\min} & \leq & \mathbf{c}_I(\mathbf{Z}) & \leq & \mathbf{c}_{I,\max}. \end{array} \tag{24}$$

It is important to note in Equations (23) and (24) that the column vector $\mathbf{z} \in \mathbb{R}^{n_z}$ contains the NLP decision variables for the *entire* problem. Similar, $\mathbf{c}(\mathbf{z})$ is a column vector of constraints for the *entire* problem. Finally the subscripts "E" and "I" correspond, respectively, to the *equality* and the *inequality* constraints. Recalling that the entire problem consists of P phases and denoting the decision vector within a given phase $p \in [1, \ldots, P]$ by $\mathbf{z}$, the complete vector of decision variables is a concatenation of the decision variables in each phase of the problem, that is,

$$\mathbf{Z} = \begin{bmatrix} \mathbf{z}^{(1)} \\ \vdots \\ \mathbf{z}^{(P)} \end{bmatrix}. \tag{25}$$

In order to make the algorithm somewhat more manageable, the constraints are rearranged such that the constraints in each phase are a concatenation of the equality constraints and inequality constraints in the phase, that is,

$$\mathbf{c}^{(p)} = \begin{bmatrix} \mathbf{c}_E^{(p)} \\ \mathbf{c}_I^{(p)} \end{bmatrix}, \tag{26}$$

where $\mathbf{c}$ is the subvector of constraints in phase $p = [1, \ldots, P]$. Finally, appended to the phase constraints are the *linkage* constraints, denoted $\mathbf{c}_L$. The total constraint vector is then given as

$$\mathbf{c} = \begin{bmatrix} \mathbf{c}^{(1)} \\ \vdots \\ \mathbf{c}^{(P)} \\ \mathbf{c}_L \end{bmatrix}. \tag{27}$$

4.1 Mapping of Decision Variables in a Single Phase

We now proceed to describe the mapping of the optimization vector within a particular phase $p \in [1, \ldots, P]$ to the values of the state, control, time, and static parameters. In order to make it easier to follow the details, the superscript $(\cdot)^{(p)}$ (denoting the phase number) is for the most part suppressed.

The total vector of optimization variables $\mathbf{z} \in \mathbb{R}^{n_x(N+2)+n_uN+n_q+2}$ in a particular phase $p = [1, \ldots, P]$ is given as

$$\mathbf{z}^{(p)} \equiv \mathbf{z} = \begin{bmatrix} \mathbf{z}_x \\ \mathbf{z}_u \\ \mathbf{q} \\ t_0 \\ t_f \end{bmatrix} \qquad (p = 1, \ldots, P), \tag{28}$$

where $\mathbf{z}_x$ is the vector of variables associated with the values of the state at the discretization points, $\mathbf{z}_u$ is the vector of variables associated with the values of the control at the LG points, $\mathbf{q}$ is the vector of static optimization parameters, t_0 is the initial time, and t_f is the terminal time. The vector $\mathbf{z}_x$ is given as

$$\mathbf{z}_x = \begin{bmatrix} \chi_1 \\ \vdots \\ \chi_{n_x} \end{bmatrix}, \tag{29}$$

where the column vectors χ_j $(j = 1, \ldots, n_x)$ are given as

$$\chi_j = \begin{bmatrix} x_{j1} \\ \vdots \\ x_{j,(N+2)} \end{bmatrix} \qquad (j = 1, \ldots, n_x), \tag{30}$$

and x_{jk} $(j = 1, \ldots, n_x; k = 1, \ldots, N + 2)$ is the value of the jth component of the state at the kth discretization point. Next, the vector $\mathbf{z}_u$ is given as

$$\mathbf{z}_u = \begin{bmatrix} \sigma_1 \\ \vdots \\ \sigma_{n_u} \end{bmatrix}, \tag{31}$$

where the column vectors σ_j $(j = 1, \ldots, n_u)$ are given as

$$\sigma_j = \begin{bmatrix} u_{j1} \\ \vdots \\ u_{j,N} \end{bmatrix} \qquad (j = 1, \ldots, n_u), \tag{32}$$

and u_{jk} $(j = 1, \ldots, n_u; k = 1, \ldots, N)$ is the value of the jth component of the control at the kth collocation (LG) point. For use in vectorized operations in MATLAB [The Mathworks, Inc. 2008], it is convenient to reshape the vectors $\mathbf{z}_x$ and $\mathbf{z}_u$ into matrices $\mathbf{Y}_x \in \mathbb{R}^{(N+2)\times n_x}$ and $\mathbf{Y}_x \in \mathbb{R}^{N\times n_u}$ whose rows contain a single vector value of the state and control, respectively. First, $\mathbf{Y}_x$ is obtained from $\mathbf{z}_x$ as

$$\mathbf{Y}_x = \mathbf{reshape}(\mathbf{z}_x, N + 2, n_x), \tag{33}$$

where **reshape** is a pseudocoded version of the MATLAB "reshape" command. The vector $\mathbf{z}_x$ can be reacquired from $\mathbf{Y}_x$ as

$$\mathbf{z}_x = \mathbf{Y}_x(:), \tag{34}$$

where ":" is a pseudocoded version of the MATLAB [The Mathworks, Inc. 2008] column-stacking operator. Next, $\mathbf{Y}_u$ is obtained from $\mathbf{z}_u$ as

$$\mathbf{Y}_u = \mathbf{reshape}(\mathbf{z}_u, N, n_u). \tag{35}$$

The vector $\mathbf{z}_u$ can be reacquired from $\mathbf{Y}_u$ as

$$\mathbf{z}_u = \mathbf{Y}_u(:). \tag{36}$$

4.2 Construction of Total Vector of Decision Variables

To construct the total vector of NLP decision variables, the process described in the previous subsection is repeated for all phases $p = [1, \ldots, P]$. In other words, we construct the vector $\mathbf{z}$ by stacking the variables $\mathbf{z}^{(p)}$ using Equation (25).

4.3 Construction of Constraint Vector within a Single Phase

Next, the algorithm for constructing the column vector of all NLP constraints within a given phase is described. The constraints within a single phase are obtained by stacking the collocated dynamic constraints, the collocated path constraints, and the boundary conditions.[6] The vector of constraints within a single phase can then be written as

$$\mathbf{c}(\mathbf{z}) = \begin{bmatrix} \mathbf{c}_E(\mathbf{z}) \\ \mathbf{c}_I(\mathbf{z}) \end{bmatrix}, \tag{37}$$

where, as before, the subscripts "E" and "I" correspond to *equality* and *inequality* constraints, respectively. A relatively straightforward way to map the equality constraints to a single column vector is as follows. First, suppose we define the *defect constraints*, $\boldsymbol{\delta}_k,\ k = 1, \ldots, N$, as

$$\boldsymbol{\delta}_k = \sum_{i=0}^{N} \mathbf{D}_{ki}\mathbf{X}_i - \frac{t_f - t_0}{2}\mathbf{f}(\mathbf{X}_k,\ \mathbf{U}_k, t_k; \mathbf{q}) \qquad (k = 1, \ldots, N), \tag{38}$$

$$\boldsymbol{\delta}_{N+1} = \mathbf{X}_{N+1} - \mathbf{X}_0 - \sum_{i=0}^{N}\sum_{k=1}^{N} w_k \mathbf{D}_{ki}\mathbf{X}_i = \mathbf{0}, \tag{39}$$

where we have assumed that $\mathbf{f}(\mathbf{X}_k,\ \mathbf{U}_k, t_k; \mathbf{q})$ is a *row* vector of length n. Then we can define the matrix $\boldsymbol{\Delta}$ as

$$\boldsymbol{\Delta} = \begin{bmatrix} \boldsymbol{\delta}_1 \\ \vdots \\ \boldsymbol{\delta}_{N+1} \end{bmatrix}. \tag{40}$$

It turns out that $\boldsymbol{\Delta}$ can be computed as

$$\boldsymbol{\Delta} = \begin{bmatrix} \mathbf{D}\mathbf{Y}_x(1 : N+1, :) - \frac{t_f - t_0}{2}\mathbf{F} \\ \mathbf{Y}_x(\text{end}, :) - \mathbf{Y}_x(1, :) - \mathbf{w}^T\mathbf{D}\mathbf{Y}(1 : N+1, :) \end{bmatrix}, \tag{41}$$

[6]It is noted that the collocated dynamic constraints are *equality* constraints while the path constraints and boundary conditions are *inequality* constraints.

where $\mathbf{w}$ is a column vector of LG weights, the notation "$1 : N + 1$" is a pseudocode for the first $N + 1$ rows of the matrix $\mathbf{Y}_x$, and $\mathbf{F}$ is a vectorized evaluation of the right-hand sides of the differential equations at the LG (collocation) points, that is,

$$\begin{aligned}\mathbf{F} &= \begin{bmatrix} \mathbf{f}(\mathbf{X}_1,\ \mathbf{U}_1, t_1; \mathbf{q}) \\ \vdots \\ \mathbf{f}(\mathbf{X}_N,\ \mathbf{U}_N, t_N; \mathbf{q}) \end{bmatrix} \\ &\equiv \mathbf{F}(\mathbf{Y}_x(2 : N + 1, :), \mathbf{Y}_u, \mathbf{t}(2 : N + 1); \mathbf{q}),\end{aligned} \tag{42}$$

where the column vector $\mathbf{t} \in \mathbb{R}^{N+2}$ is given as

$$\mathbf{t} = \begin{bmatrix} t_0 \\ t_1 \\ \vdots \\ t_{N+1} \end{bmatrix}. \tag{43}$$

The matrix $\mathbf{\Delta}$ can then be isomorphically mapped to the column vector of equality constraints $\mathbf{c}_E$ using a pseudocoded version of the MATLAB [The Mathworks, Inc. 2008] column-stacking operator ":" as

$$\mathbf{c}_E = \mathbf{\Delta}(:). \tag{44}$$

Next, let $\mathbf{H}_C$ be the matrix that arises from the evaluation of the path constraints at all of the LG points. Then $\mathbf{H}$ is given as

$$\mathbf{H}_C = \begin{bmatrix} \mathbf{C}(\mathbf{X}_1,\ \mathbf{U}_1, t_1; \mathbf{q}) \\ \vdots \\ \mathbf{C}(\mathbf{X}_N,\ \mathbf{U}_N, t_N; \mathbf{q}) \end{bmatrix} \equiv \mathbf{C}(\mathbf{Y}_x(2 : N + 1, :), \mathbf{Y}_u, \mathbf{t}; \mathbf{q}). \tag{45}$$

Also, let $\mathbf{H}_\phi$ be the vector that arises from the evaluation of the boundary conditions. Noting that the boundary conditions are functions of only the state and time at the endpoints of the phase, we have

$$\mathbf{H}_\phi = \phi(\mathbf{Y}_x(1, :), t_0, \mathbf{Y}_x(\text{end}, :), t_f), \tag{46}$$

where it is noted that $\mathbf{Y}_x(1, :) = \mathbf{X}_0$ and $\mathbf{Y}_x(\text{end}, :) = \mathbf{X}_f$, respectively. The inequality constraint vector is then given as

$$\mathbf{c}_I = \begin{bmatrix} \mathbf{H}_C(:) \\ \mathbf{H}_\phi \end{bmatrix}, \tag{47}$$

where we once again note the use of the MATLAB [The Mathworks, Inc. 2008] column-stacking operator ":" to isomorphically map the matrix of path constraints to a column vector of length $n_c N$.

4.4 Construction of Linkage Constraint Vector

The vector of linkage constraints is obtained by using the values of the state and time at the terminus and start of each pair that is to be linked. We can

write the evaluation of the sth set of linkage constraints as follows:

$$\begin{matrix} t_f^{(p_l^{(s)})} - t_0^{(p_r^{(s)})} \\ \mathbf{S}^{(s)}\left(\mathbf{Y}_x^{(p_l^{(s)})}(\text{end}, :), \mathbf{q}^{(p_l^{(s)})}, \mathbf{Y}_x^{(p_r^{(s)})}(1, :), \mathbf{q}^{(p_r^{(s)})}\right) \end{matrix} \qquad \begin{cases} p_l, p_r \in [1, \ldots, P], \\ s = 1, \ldots, L. \end{cases} \tag{48}$$

It is noted that the linkage conditions in Equation (48) have been separated into those that link the independent variable (i.e., time) between phases and those that link the state and parameters between phases. The reason for this separation is that the linkage of the independent variable is a *linear* constraint and thus has a structure that can be taken advantage of when implementing the NLP in software.

5. CONSTRUCTION OF TOTAL VECTOR OF CONSTRAINTS

The total vector of constraints for all phases and the linkage constraints is then given as

$$\mathbf{c}(\mathbf{Z}) = \left\{ \begin{matrix} \begin{bmatrix} \boldsymbol{\Delta}^{(1)}(:) \\ \mathbf{H}_C^{(1)}(:) \\ \mathbf{H}_\phi^{(1)} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \boldsymbol{\Delta}^{(P)}(:) \\ \mathbf{H}_C^{(P)}(:) \\ \mathbf{H}_\phi^{(P)} \end{bmatrix} \\ \begin{bmatrix} t_f^{(p_l^{(1)})} - t_0^{(p_r^{(1)})} \\ \mathbf{S}^{(1)}\left(\mathbf{Y}_x^{(p_l^{(1)})}(\text{end}, :), \mathbf{q}^{(p_l^{(1)})}, \mathbf{Y}_x^{(p_r^{(1)})}(1, :), \mathbf{q}^{(p_r^{(1)})}\right) \end{bmatrix} \\ \vdots \\ \begin{bmatrix} t_f^{(p_l^{(L)})} - t_0^{(p_r^{(L)})} \\ \mathbf{S}^{(L)}\left(\mathbf{Y}_x^{(p_l^{(L)})}(\text{end}, :), \mathbf{q}^{(p_l^{(L)})}, \mathbf{Y}_x^{(p_r^{(L)})}(1, :), \mathbf{q}^{(p_r^{(L)})}\right) \end{bmatrix} \end{matrix} \right\}. \tag{49}$$

6. STRUCTURE OF GAUSS PSEUDOSPECTRAL NLP SPARSITY PATTERN

As it turns out, the NLP described in the previous section has an extremely well-defined structure that can be taken advantage of in a software implementation. First, the constraint Jacobian, $\partial\mathbf{c}/\partial\mathbf{z}$, is *sparse*. A graphical representation of the structure of a single phase of the constraint Jacobian is shown in Figure 2. The sparsity pattern for a given phase $p \in [1, \ldots, P]$ is divided into rows and columns such that the rows correspond to the phase constraints while the columns correspond to the phase variables. Furthermore, the rows in Figure 2 contain the derivatives of the defect constraints, path constraints, and

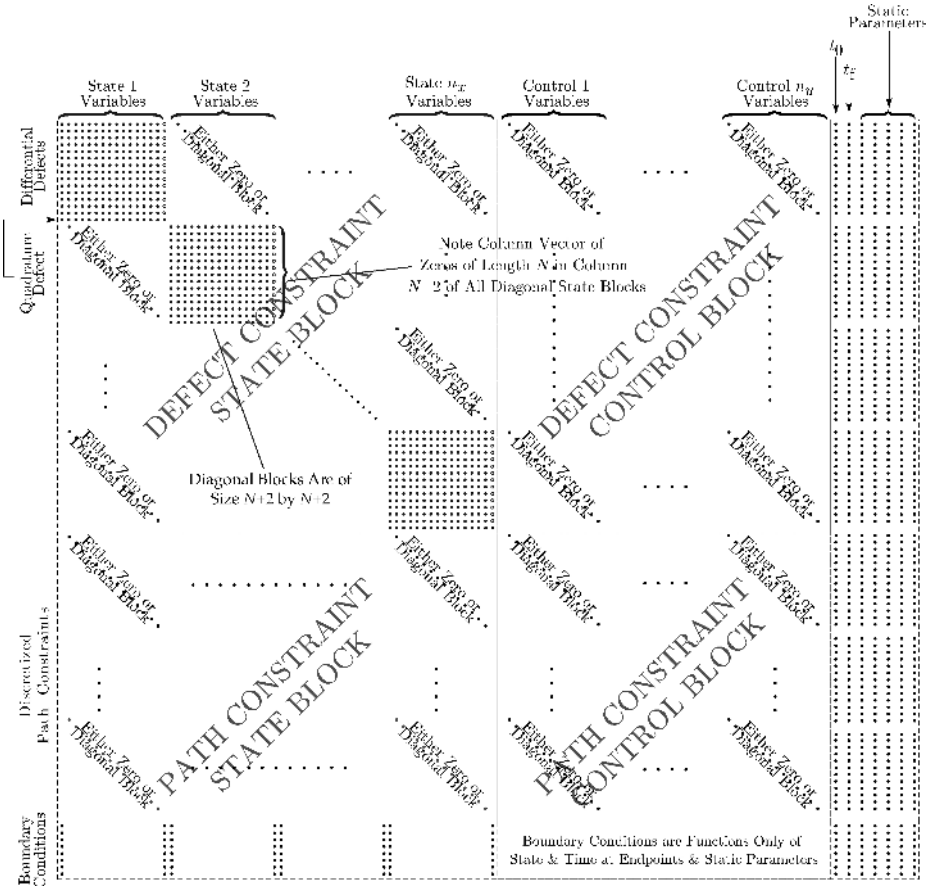

Fig. 2. Structure of single phase of constraint Jacobian of discretization of multiple-phase Gauss pseudospectral method.

event constraints with respect to each value of the state, control, initial and terminal times, and the static parameters. The derivatives of the defect constraints are divided into a set of main diagonal blocks and off-diagonal blocks. The main diagonal blocks are the derivatives of the ith set of defect constraints (i.e., those associated with the ith state) with respect to the values of the ith state at the discretization points. The off-diagonal blocks of the derivatives of the defect constraints are the derivatives with respect to the values of the jth state (where $j \neq i$) and the kth control (in that order column-wise). Finally, the columns that lie to the right of those associated with the control are the derivatives with respect to the initial time, t_0, the terminal time, t_f, and the static parameters $\mathbf{q}$, respectively. The rows of the phase sparsity that lie below those associated with the defect constraints are the derivatives of the path constraints and contain the derivatives with respect to the values of the state, control, initial and terminal times, and static parameters. Finally, the rows that lie below those associated with the path constraints contain the derivatives of the event

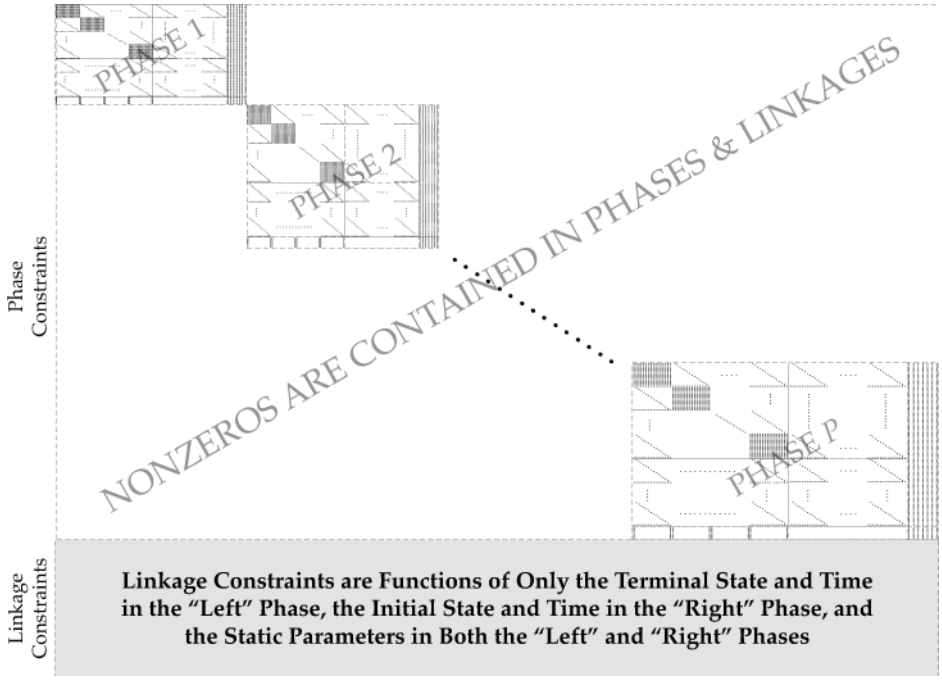

Fig. 3. Structure of entire sparsity pattern for discretization of the multiple-phase Gauss pseudospectral Method. Each phase has nonzeros as given in Figure 2.

constraints. It is seen that the event constraint derivatives are nonzero only with respect to the initial state and terminal state, the initial and terminal time, and the static parameters.

Using the phase sparsity given in Figure 2, the complete sparsity pattern for a multiple-phase problem is shown in Figure 3. It is seen that the complete sparsity pattern is block-diagonal concatenation of the sparsity pattern in each phase, where the variables in each phase are nonoverlapping with the exception of the linkage constraints. With regard to the linkage constraint derivatives, the overlap in variables between phases is confined to variables at the terminus and start of the pair of phases to be linked.

7. LAGRANGE MULTIPLIERS OF NLP AND COSTATE MAPPING

Each constraint in the NLP has a corresponding Lagrange multiplier. Suppose we let $\tilde{\Lambda}_k^{(p)}(k = 1, \ldots, N)$ and $\tilde{\Lambda}_f^{(p)}$ be the Lagrange multipliers associated with the discretized dynamic constraints of Equation (16) and the quadrature constraint of Equation (19), respectively, in each phase $p = [1, \ldots, P]$ of the optimal control problem. Furthermore, let $\tilde{\mu}_k(k = 1, \ldots, N)$ be the Lagrange multipliers associated with the discretized path constraints of Equation (20). Finally, let $\tilde{\nu}$ be the Lagrange multiplier associated with the discretized boundary conditions of Equation (21). Then it has been shown [Benson 2004; Benson et al. 2006; Huntington 2007] that the costate (i.e., the adjoint) of the continuous-time

optimal control problem is related to the Lagrange multipliers $\tilde{\mathbf{\Lambda}}_k (k = 1, \ldots, N)$ and $\tilde{\mathbf{\Lambda}}_f$ as

$$\begin{aligned}
\mathbf{\Lambda}(t_f) &= \tilde{\mathbf{\Lambda}}_f, \\
\mathbf{\Lambda}_k &= \tilde{\mathbf{\Lambda}}_k / w_k, \\
\mathbf{\Lambda}(t_0) &= (1+\alpha)\tilde{\mathbf{\Lambda}}(t_f) - \sum_{i=1}^{N} D_{i0}\tilde{\mathbf{\Lambda}}_i, \\
\boldsymbol{\mu}_k &= \frac{2}{t_f - t_0}\frac{\tilde{\boldsymbol{\mu}}_k}{w_k}, \\
\boldsymbol{\nu} &= \tilde{\boldsymbol{\nu}}, \\
\alpha &= \sum_{i=1}^{N} w_i D_{i0}.
\end{aligned} \tag{50}$$

The NLP described in Section 4 has a dual solution that includes the Lagrange multipliers for each category of discretized constraints. In particular, suppose we let $\mathbf{\Gamma}$ be the vector of Lagrange multipliers associated with the differential dynamic constraints and the quadrature constraints. Then, because the dynamics are discretized at N Legendre-Gauss points, we know that $\mathbf{\Gamma} \in \mathbb{R}^{n_x(N+1)}$ (i.e., we have $n_x N$ multipliers for the discretized dynamic constraints and another N multipliers for the quadrature constraints). The column vector $\mathbf{\Gamma}$ can be isomorphically mapped to a matrix of size $(N+1) \times n_x$ via a pseudocoded version of the MATLAB [The Mathworks, Inc. 2008] command **reshape** [The Mathworks, Inc. 2008] as

$$\tilde{\mathbf{M}} = \mathbf{reshape}(\mathbf{\Gamma}, N+1, n_x) = \begin{bmatrix} \tilde{\mathbf{\Lambda}}_1 \\ \tilde{\mathbf{\Lambda}}_2 \\ \vdots \\ \tilde{\mathbf{\Lambda}}_{N+1} \end{bmatrix}, \tag{51}$$

where $\tilde{\mathbf{\Lambda}}_i (i = 1, \ldots, N+1)$ are row vectors. Next, we can construct a diagonal matrix $\mathbf{W}$ that contains the reciprocals of the Gauss weights as

$$\mathbf{W} = \mathbf{diag}(1/w_1, \ldots, 1/w_N). \tag{52}$$

The costate estimate given in Equation (50) can then be obtained at the Legendre-Gauss points as

$$\mathbf{M} = \mathbf{W}\tilde{\mathbf{M}}(1 : \text{end} - 1, :). \tag{53}$$

In addition, the costate estimate at the *initial time* in phase $p = [1, \ldots, P]$ is computed as

$$\mathbf{\Lambda}(t_0) = (1 + \mathbf{w}^T \mathbf{D}(:, 1))\tilde{\mathbf{\Lambda}}_f - \mathbf{sum}(\mathbf{repmat}(\mathbf{D}(:, 1), 1, n_x) .\mathbf{M}(1 : \text{end} - 1, :), 1), \tag{54}$$

where "." denotes element-by-element multiplication and we have used pseudocoded versions of the MATLAB commands **repmat** and **sum** (and, by virtue

of the last argument being unity, the sum is taken along the columns). We can then augment the initial costate to the matrix $\mathbf{M}$ to obtain the complete matrix of costates as

$$\mathbf{M}_a = \begin{bmatrix} \mathbf{\Lambda}_0 \\ \mathbf{\Lambda}_1 \\ \mathbf{\Lambda}_2 \\ \vdots \\ \mathbf{\Lambda}_{N+1} \end{bmatrix}. \tag{55}$$

8. SEPARATION OF CONSTANT AND NONCONSTANT DERIVATIVES IN CONSTRAINT JACOBIAN

A unique feature of the *SNOPTA* interface of the NLP solver SNOPT [Gill et al. 2002] is that SNOPTA allows the user to separate the constant derivatives from the nonconstant derivatives. This feature is described in the SNOPT 7 manual [Gill et al. 2006] but is repeated here in the context of the Gauss pseudospectral method. First, consider a problem such that each function (constraint or objective) can be written in the form

$$F_i(\mathbf{x}) = f_i(\mathbf{x}) + \sum_{j=1}^{n} A_{ij}\mathbf{x}_j. \tag{56}$$

$f_i(\mathbf{x})$ is a nonlinear function, n is the total number of optimization (decision) variables, $\mathbf{x} \in \mathbb{R}^n$, and the elements A_{ij} are *constant*. Clearly the Jacobian of $\mathbf{F}$ is given as

$$\frac{\partial \mathbf{F}}{\partial \mathbf{x}} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} + \mathbf{A}, \tag{57}$$

where

$$\mathbf{A} = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & A_{22} & \cdots & A_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m1} & A_{m2} & \cdots & A_{mn} \end{bmatrix}. \tag{58}$$

Suppose now that there exist elements in the matrix $\mathbf{A}$ such that the corresponding elements in the matrix $\partial\mathbf{f}/\partial\mathbf{x}$ is *zero*. In other words, suppose that there are derivatives of $\mathbf{F}$ that are *constant*. Then these constant derivatives can be *stored* (i.e., they never need to be computed, but can be retrieved when necessary).

The Gauss pseudospectral method is an example of a situation where the exploitation of separability can be of great benefit. Specifically, it is known that the majority of the derivatives in the differential defect constraints are constant. In particular, the only nonconstant elements in a main-diagonal block are the N elements that contain the derivatives of the right-hand side of the differential equations. All other elements in a main-diagonal block are constant. A schematic of the structure of the constant and nonconstant elements of a

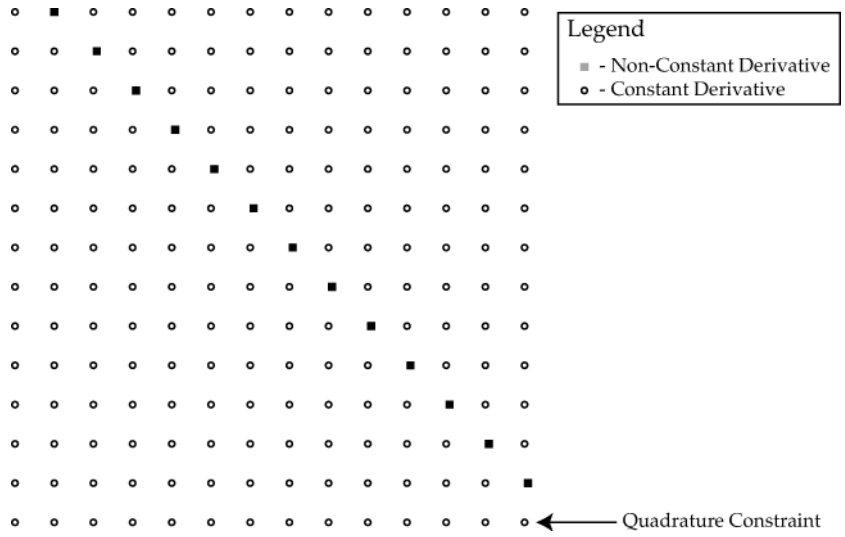

Fig. 4. Schematic showing constant and nonconstant derivatives elements of a main-diagonal block of the differential equation defect constraints.

main-diagonal block is given in Figure 4. Moreover, the sparsity pattern for all of the nonconstant elements in a particular phase is given in Figure 5.

9. LINEAR CONSTRAINTS

As mentioned earlier, the constraints on the independent variable are *linear*. These linear constraints fall into two categories: monotonicity constraints and linkage constraints. First, we know that the independent variable must be *monotonic* (i.e., either increasing or decreasing throughout the problem). In order to ensure monotonicity, the following constraints must be placed on the independent variable in each phase $p \in [1, \ldots, P]$ of the problem:

$$\begin{aligned} t_f - t_0 &\geq 0 \quad \text{for an increasing independent variable} \\ t_f - t_0 &\leq 0 \quad \text{for a decreasing independent variable} \end{aligned} \qquad (p \in [1, \ldots, P]). \tag{59}$$

Next, the independent variable must be *continuous* at a phase interface. The continuity conditions on the independent variable are then restated from

$$t_f^{(p_l^{(s)})} - t_0^{(p_r^{(s)})} \qquad (p_l, p_r \in [1, \ldots, L]), \tag{60}$$

where we recall that L is the number of linkage pairs. The entire set of linear constraints can then be written in the general linear constraint form

$$\mathbf{L}_{\min} \leq \mathbf{Bz} \leq \mathbf{L}_{\max}, \tag{61}$$

where $\mathbf{B} \in \mathbb{R}^{n_l \times n_z}$ is matrix of coefficients for the linear constraints (in this case a matrix containing only zero, -1, and 1), $\mathbf{L}_{\min} \in \mathbb{R}^{n_l}$ are the lower bounds on the linear constraints, and $\mathbf{L}_{\max} \in \mathbb{R}^{n_l}$ are the upper bounds on the linear constraints. It is noted that the lower and upper bounds on the linear constraints

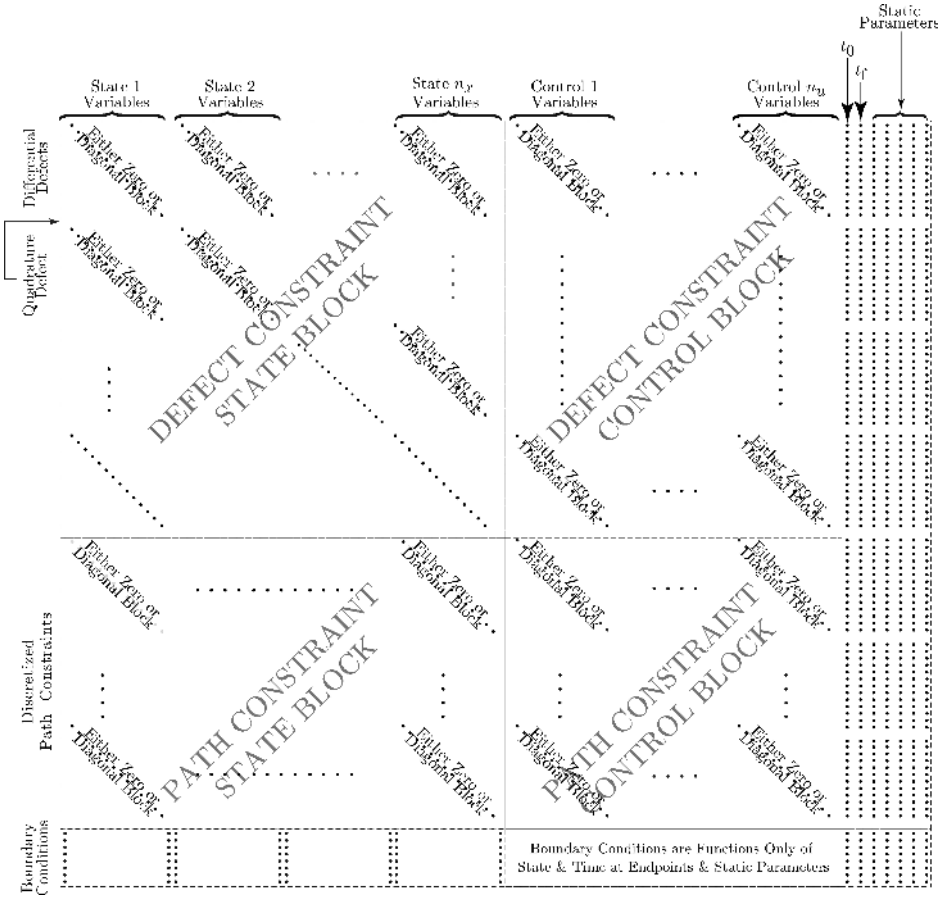

Fig. 5. Schematic showing the nonconstant derivatives in a single phase of an optimal control problem discretized via the Gauss pseudospectral method.

are obtained directly from Equations (59) and (60), respectively. Finally, when treating the constraints on the independent variable as linear, these equations are removed from the general constraint vector given in Equation (49).

10. COMPUTATION OF LEGENDRE-GAUSS POINTS, WEIGHTS, AND DIFFERENTIATION MATRIX

A key part of the algorithm described in this article is the ability to compute the Legendre-Gauss points, weights, and differentiation matrix. It is noted that the Legendre-Gauss points are the roots of the Nth degree Legendre polynomial while the Legendre-Gauss weights are computed using Equation (9). A convenient way to compute the Legendre-Gauss points is via the eigenvalues of the tridiagonal *Jacobi* matrix. The Legendre-Gauss weights are then computed using Equation (9) where the derivative of the Nth-degree Legendre polynomial,

$\dot{P}_N$, is computed as

$$\dot{P}_N = \frac{-(n+1)P_N(\tau)}{1-\tau^2}. \tag{62}$$

Finally, the Gauss pseudospectral differentiation matrix (which is a nonsquare matrix $\mathbf{D} \in \mathbb{R}^{N\times(N+1)}$) is given as

$$\mathbf{D}_{ki} = \begin{cases} \dfrac{(1+\tau_k)\dot{P}_N(\tau_k) + P_N(\tau_k)}{(\tau_k - \tau_i)\left[(1+\tau_i)\dot{P}_N(\tau_i) + P_N(\tau_i)\right]} & i \neq k, \\ \dfrac{(1+\tau_i)\ddot{P}_N(\tau_i) + 2\dot{P}_N(\tau_i)}{2\left[(1+\tau_i)\dot{P}_N(\tau_i) + P_N(\tau_i)\right]} & i = k. \end{cases} \tag{63}$$

11. COMPUTATION OF ENDPOINT CONTROLS

Because solving the NLP from the Gauss pseudospectral method results in controls at only at the the interior points (i.e., the Legendre-Gauss points), it is necessary to obtain the endpoint controls after the NLP is solved. In the algorithm presented here the endpoint controls are computed in two different ways. The first method for computing the endpoint control is simply to extrapolate the control to the initial and terminal times. A second method for endpoint control computation is to employ the Pontryagin minimum principle [Kirk 2004] using the endpoint values of the state and costate. This second method requires that the following auxiliary NLP be solved. Minimize the Hamiltonian

$$H = \mathcal{L} + \boldsymbol{\lambda}^T \mathbf{f} \tag{64}$$

subject to constraint

$$\mathbf{C}_{\min} \leq \mathbf{C} \leq \mathbf{C}_{\max}. \tag{65}$$

The optimization problem given in Equations (64) and (65) is implemented by using the values of the state and costate at the endpoints of each phase obtained by solving the GPM NLP and minimizing over the allowable set of controls. The endpoint control optimization problem is small (having only a number of variables equal to the number of controls in each phase) and, thus, can generally be solved quickly. It is noted, however, that solving the NLP given in Equations (64) and (65) can lead to erratic controls for certain types of problems (e.g., problems with singular arcs or bang-bang controls). Thus, both methods described in this section are used for endpoint control computation and the user can choose whichever control is most suitable for a particular application.

12. AUTOMATIC SCALING OF NLP AND DETERMINATION OF DEPENDENCIES

An extremely important issue that arises in the solution of any optimization problem is *scaling*. A poorly scaled problem can lead to either extremely slow convergence or divergence [Betts 2001]. While not a great deal of information exists in the literature on scaling (primarily because scaling is more of an art

than a science), it is a practical matter that needs to be dealt with in any implementation. In this section we describe a scaling procedure [Betts 2001]. It has been found that this scaling procedure works extremely well on a wide variety of problems and can be used as a substitute for the tedious work of manual scaling. Finally, it is noted that the procedure described here scales the functions of the *optimal control problem* as opposed to scaling the functions of the NLP.

The scaling procedure used in the current algorithm is divided into two parts. The first part of the procedure automatically scales the variables based on the bounds specified by the user. Given the user-specified bounds on the time, state, control, and parameters in each phase of the problem, the unscaled NLP variables are given as box-bound constraints [see Equation (24)] as

$$\mathbf{z}_{\min} \leq \mathbf{z} \leq \mathbf{z}_{\max}. \tag{66}$$

Now let $\mathbf{S_z}$ be a diagonal matrix whose diagonal elements contain the scale factors for the variables. Then the scaled value of $\mathbf{z}$, denoted $\tilde{\mathbf{z}}$, is given as

$$\tilde{\mathbf{z}} = \mathbf{S_z}\mathbf{z}. \tag{67}$$

Because the user may set some of the lower and upper limits to $\pm\infty$, the first step is to find the infinite limits and set the diagonal elements in $\mathbf{S_z}$ to unity. Next, for the lower and upper limits that remain, the *larger* of $|\mathbf{z}_{\min}|$ and $|\mathbf{z}_{\max}|$ is determined. The corresponding diagonal elements of $\mathbf{S_z}$ are then set to the reciprocal of the larger values. In this manner, the variables are scaled such that their scaled limits either lie between -1 and 1, -1 and ∞, $-\infty$ and 1, or $-\infty$ and ∞. It is noted, however, that if sensible noninfinite bounds are chosen, all scaled limits will lie between -1 and 1.

The second step in the automatic scaling procedure is to scale the functions of the optimal control problem. First, as has been suggested in the literature [Betts 2001], the differential equation constraints (i.e., the defect constraints) are scaled using the same scale factors as used to scale the states. Next, suppose we let $\mathbf{x}$, $\mathbf{u}$, and $\mathbf{q}$ be the state, control, and static parameters in a given phase $p \in [1, \ldots, P]$ of the optimal control problem with corresponding scaled values $\tilde{\mathbf{x}}$, $\tilde{\mathbf{u}}$, and $\tilde{\mathbf{q}}$. Furthermore, let $\mathbf{S_x}$, $\mathbf{S_u}$, and $\mathbf{S_q}$ be diagonal matrices whose diagonal elements contain the corresponding scale factors. We then have

$$\begin{aligned} \tilde{\mathbf{x}} &= \mathbf{S_x}\mathbf{x}, \\ \tilde{\mathbf{u}} &= \mathbf{S_u}\mathbf{u}, \\ \tilde{\mathbf{q}} &= \mathbf{S_q}\mathbf{q}. \end{aligned} \tag{68}$$

Finally, let $\tilde{\mathbf{C}}$ be the scaled value of the path constraints with a corresponding diagonal scaling matrix $\mathbf{S_C}$. Then, in a manner similar to the variables, we have

$$\tilde{\mathbf{C}} = \mathbf{S_C}\mathbf{C}. \tag{69}$$

Suppose now that the Jacobian of $\mathbf{C}$ is computed as

$$\left[\frac{\partial \mathbf{C}}{\partial \mathbf{x}} \quad \frac{\partial \mathbf{C}}{\partial \mathbf{u}} \quad \frac{\partial \mathbf{C}}{\partial \mathbf{q}} \right]. \tag{70}$$

Then the scale factors for the path constraints are obtained by (1) evaluating the Jacobian given in Equation (70) at a specified number of trial points $(\mathbf{x}_i, \mathbf{u}_i, \mathbf{q}_i)$ $(i = 1, \ldots, n_T)$, where n_T is the number of trials in the phase; (2) computing the row norm of the path constraint Jacobian at the trial points; and (3) taking the average of the row norms. The scale factor for each scalar path constraint is then applied at each collocation point. The boundary conditions (i.e., the event constraints) and the linkage constraints are scaled in a manner similar to that used to scale the differential-algebraic equations.

Simultaneous with the determination of scale factors for the NLP, a procedure has been implemented to determine the dependencies of the differential equations and path constraints on the state and control. In particular, recall in Section 6 that the off-diagonal blocks in each phase are either zero or structured depending upon whether a particular differential equation or path constraint is a function of a particular variable. In the algorithm described here, the Jacobian *pattern* (i.e., a matrix of ones and zeros) is determined at the n_T trial points. If at any of the trial points a particular element in the Jacobian is nonzero, then this dependence is included in the NLP sparsity pattern and the particular off-diagonal block in Figure 5 is nonzero.

13. NUMERICAL METHODS FOR COMPUTATION OF OBJECTIVE FUNCTION GRADIENT AND CONSTRAINT JACOBIAN

A key issue that arises in the solution of any NLP is the computation of the derivatives to obtain the objective function gradient and constraint Jacobian. GPOPS has three options for computation of these quantities. The first option is the built-in finite difference method in SNOPT [Gill et al. 2006]. While finite differencing will work on some problems, it is computationally inefficient and can be problematic due to inaccurate derivative approximations, thereby leading to a failure of SNOPT to converge to an optimal solution. As a result, the following derivative methods can also be used in GPOPS:

(1) Automatic differentiation via one of the following programs:
   (a) built-in forward mode automatic differentiation,
   (b) *Matlab Automatic Differentiation (MAD)* [Forth 2006; Forth and Edvall 2007],
   (c) *Interval Laboratory (INTLAB)* [Rump 2008];
(2) complex-step differentiation [Martins et al. 2003];
(3) analytic (user-supplied) derivatives.

We have provided three options for automatic differentiation primarily because some users may prefer to obtain a commercial automatic differentiation package *(MAD)* with the maximum functionality while others may be prefer a freely available package (*INTLAB* or the built-in automatic differentiator). Next, as discussed in Martins et al. [2003], complex-step differentiation provides extremely high-accuracy derivatives and is insensitive to the choice of the step size (as compared with finite difference approximations which create round-off error for small values of the step). It is noted that the current procedure for complex-step differentiation is quite simple in that it does independently

perturb variables in a particular phase (which is possible in this case due to the fact, in any given phase, the constraints are functions of variables in only that phase and are independent of variables in the other phases).[7] Furthermore, the following functions in MATLAB can create issues with complex-step differentiation: "abs," "min," "max," and "dot."[8] Finally, the user needs to code the problem carefully and not use the standard transpose operation in MATLAB, but use the "dot-transpose" to ensure that a *real transpose* is taken (i.e., not a complex-conjugate transpose). Finally, the complex-step method has been tested on all of the problems included with the GPOPS software and has been found to work extremely well in practice.

14. USER INTERFACE TO GPOPS

GPOPS has a user interface that enables the optimal control problem to be input in an intuitive yet compact manner. The key MATLAB programming element that makes this user interface possible is the structure. In particular, GPOPS utilizes multilevel structures that enable compact specification of the lower and upper bounds on all variables and constraints in each phase of the problem and provide equal flexibility when specifying how the phases are to be linked. Finally, all of the functions in the optimal control problem (i.e., cost function, differential-algebraic equations, boundary conditions, and linkage conditions) use structure inputs, thus being consistent with other inputs in the software. All of the details regarding the GPOPS interface are found in the GPOPS user's manual that accompanies this article.

15. APPLICATIONS OF GPOPS

In this section we consider three examples that demonstrate various aspects of the GPOPS software. The first example is a modified version of the well-known chemical engineering problem called the *Lee-Ramirez bioreactor* where it is desired to maximize profitability of a fed-batch reactor for induced foreign protein production by recombinant bacteria [Balsa-Canto et al. 2001]. The second and third problems are aerospace engineering applications of maximizing the final mass during the ascent of a multiple-stage launch vehicle [Benson 2004] and maximizing the final altitude during a one-dimensional ascent of a sounding rocket [Betts 2001], respectively. As mentioned earlier, the MATLAB version of the sparse NLP solver SNOPT [Gill et al. 2006] was used. Moreover, in all examples the default optimization and feasibility tolerances were used in SNOPT along with a limited memory Hessian and an LU factorization with complete pivoting (see the SNOPT manual [Gill et al. 2006] for more details). All examples given below are included in the software distribution of GPOPS.

[7]It is noted that algorithms such as that given in Curtis et al. [1974] perturb groups of variables, and such an algorithm is expected to be implemented in a future version of GPOPS.

[8]Complex-step differentiation of the functions "abs," "min," and "max" can be accommodated by developing a MATLAB class, which is likely to be implemented in a future release of GPOPS.

15.1 Example 1: Modified Lee-Ramirez Bioreactor Problem

Consider the following optimal control problem. Maximize the cost functional

$$J = x_1(t_f)x_4(t_f) \tag{71}$$

subject to the dynamic constraints

$$\begin{aligned}
\dot{x}_1 &= u_1 + u_2, \\
\dot{x}_2 &= g_1 x_2 - \frac{u_1 + u_2}{x_1} x_2, \\
\dot{x}_3 &= c_1 \frac{u_1}{x_1} - \frac{u_1 + u_2}{x_1} x_3 - g_1 \frac{x_2}{c_2}, \\
\dot{x}_4 &= g_2 x_2 - (u_1 + u_2)\frac{x_4}{x_1}, \\
\dot{x}_5 &= c_3 \frac{u_2}{x_1} - \frac{u_1 + u_2}{x_1} x_5, \\
\dot{x}_6 &= -g_3 x_6, \\
\dot{x}_7 &= g_3(1 - x_7),
\end{aligned} \tag{72}$$

where x_1 is the reactor volume, x_2 is the cell density, x_3 is the nutrient concentration, x_4 is the foreign protein concentration, x_5 is the inducer concentration, x_6 is the inducer shock factor on cell growth rate, x_7 is the inducer recovery factor on cell growth rate, u_1 is the glucose rate, and u_2 is the inducer feeding rate. The coefficients g_1, g_2, and g_3 are given as follows:

$$\begin{aligned}
t_1 &= 14.35 + x_3 + \frac{x_3^2}{111.5}, \\
t_2 &= 0.22 + x_5, \\
t_3 &= x_6 + \frac{0.22}{t_2} x_7, \\
g_1 &= \frac{x_3}{t_1}\left[x_6 + 0.22\frac{x_7}{t_2}\right], \\
g_2 &= 0.233 \frac{x_3}{t_1} \frac{0.0005 + x_5}{0.022 + x_5}, \\
g_3 &= 0.09 \frac{x_5}{0.034 + x_5}.
\end{aligned} \tag{73}$$

It is noted that $(x_1, \ldots, x_7)$ are the components of the state while (u_1, u_2) are the components of the control for this problem. The controls are constrained as

$$\begin{aligned}
0 \le u_1 \le 1, \\
0 \le u_2 \le 1.
\end{aligned} \tag{74}$$

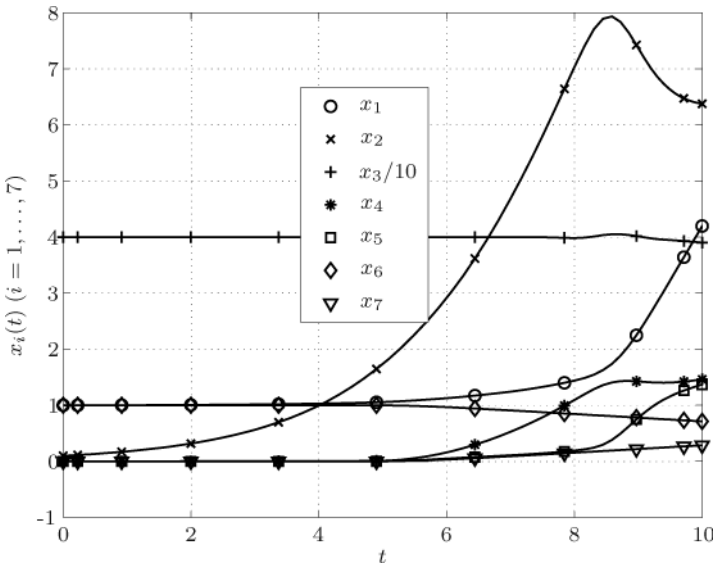

Fig. 6. Components of the state for the modified Lee-Ramirez bioreactor problem.

For this example we choose a fixed final time of $t_f = 10$ with the initial conditions

$$\begin{aligned} x_1(0) &= 1, & x_2(0) &= 0.1, \\ x_3(0) &= 40, & x_4(0) &= 0, \\ x_5(0) &= 0, & x_6(0) &= 1, \\ & & x_7(0) &= 0. \end{aligned} \tag{75}$$

Finally, it is known that the optimal control has an extremely large rate near the end of the time interval. Thus, in order to obtain a well-behaved control without greatly affecting the cost, we modify the cost function of Equation (71) as in Rutquist and Edvall [2008]:

$$J = x_1(t_f)x_4(t_f) + f \int_0^{t_f} \left(w_1^2 + w_2^2\right)dt, \tag{76}$$

where $f = 0.1N^{-1}$ [Rutquist and Edvall 2008] (N = number of LG points) is a small penalty term and w_1 and w_2 are pseudocontrols obtained by augmenting the dynamics of Equation (72) with the following two equations:

$$\begin{aligned} \dot{u}_1 &= w_1, \\ \dot{u}_2 &= w_2. \end{aligned} \tag{77}$$

As stated, the integral term in Equation (76) is a small penalty added to the cost in order to obtain a smooth control and this penalty approaches zero as N gets large.

The modified Lee-Ramirez bioreactor problem was solved in GPOPS as a one-phase optimal control problem using 80 LG points and built-in automatic differentiation. The 80 LG point solution is shown in Figures 6–8, where it

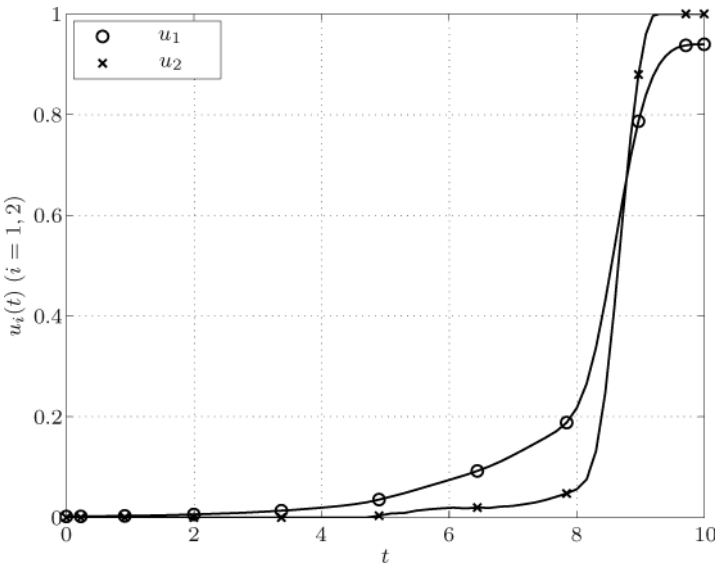

Fig. 7. Components of the control for the modified Lee-Ramirez bioreactor problem.

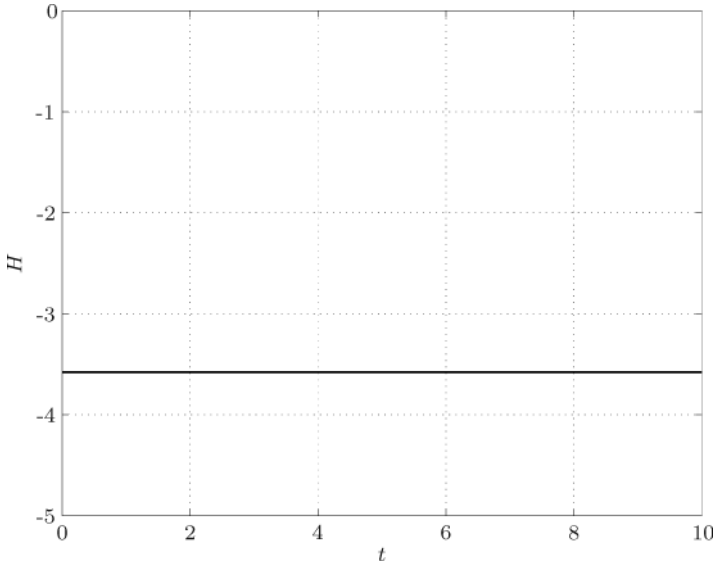

Fig. 8. Hamiltonian for the modified Lee-Ramirez bioreactor problem.

is noted that the state x_3 is divided by 10 to improve the visual appearance of Figure 6. While not shown in Figures 6–8, the optimal trajectory and control obtained using GPOPS is in excellent agreement with the corresponding solution obtained using the commercial software program *PROPT* [Rutquist and Edvall 2008] (the results of the *PROPT* solution for this problem can be

Table I. Mass and Propulsion Properties of Launch Vehicle Ascent Problem

	Solid rocket boosters	Stage 1	Stage 2
Total mass (kg)	19290	104380	19300
Propellant mass (kg)	17010	95550	16820
Engine thrust (N)	628500	1083100	110094
$I_s p$ (s)	283.3334	301.6878	467.2131
Number of engines	9	1	1
Burn time (s)	75.2	261	700

found online[9]). Moreover, the optimal objective functions obtained with GPOPS and *PROPT* are approximately 6.14923 and 6.14933, respectively. Finally, it is known theoretically that the optimal Hamiltonian for this problem is constant (because the Hamiltonian is not an explicit function of time) and Figure 8 shows that the Hamiltonian obtained from GPOPS is in excellent agreement with this theoretical result.

15.2 Example 2: Multiple-Stage Launch Vehicle Ascent Problem

The problem considered in this section is the ascent of a multiple-stage launch vehicle. The objective is to maneuver the launch vehicle from the ground to the target orbit while maximizing the remaining fuel in the upper stage. It is noted that this example is found in several places in the open literature [Benson 2004; Huntington 2007].

15.2.1 *Vehicle Properties.* The launch vehicle considered in this example has two main stages along with nine strap-on solid rocket boosters. The flight of the vehicle can be divided into *four* distinct phases. The first phase begins with the rocket at rest on the ground and, at time t_0, the main engine and six of the nine solid boosters ignite. When the boosters are depleted at time t_1, their remaining dry mass is jettisoned. The final three boosters are then ignited, and, along with the main engine, represent the thrust for the second phase of flight. These three remaining boosters are jettisoned when their fuel is exhausted at time t_2, and the main engine alone creates the thrust for the third phase. The fourth phase begins when the main engine fuel has been exhausted (MECO) and the dry mass associated with the main engine is ejected at time t_3. The thrust during phase four is from a second stage, which burns until the target orbit has been reached (SECO) at time t_4, thus completing the trajectory. The specific characteristics of these rocket motors can be seen in Table I.[10] Note that the solid boosters and main engine burn for their entire duration (meaning t_1, t_2, and t_3 are fixed), while the second stage engine is shut off when the target orbit is achieved (t_4 is free).

[9]`http://tomdyn.com`.

[10]It is noted that the values of specific impulse shown in Table I were obtained by computing $Tt_b/(g_0 m_{\text{prop}})$, where T is the thrust, t_b is the burn time, $g_0 = 9.80665$, and m_{prop} is the propellant mass

Table II. Constants Used in Launch Vehicle Example

Constant	Value
Payload Mass (kg)	4164
S (m^2)	4π
C_d	0.5
ρ_0 (kg/m^3)	1.225
H (m)	7200
t_1 (s)	75.2
t_2 (s)	150.4
t_3 (s)	261
R_e (m)	6378145 m
Ω (rad/s)	$7.29211585 \times 10^{-5}$

15.2.2 *Dynamic Model.* The equations of motion for a nonlifting point mass in flight over a spherical rotating planet are expressed in Cartesian earth-centered inertial (ECI) coordinates as

$$\begin{aligned} \dot{\mathbf{r}} &= \mathbf{v}, \\ \dot{\mathbf{v}} &= -\frac{\mu}{\|\mathbf{r}\|^3}\mathbf{r} + \frac{T}{m}\mathbf{u} + \frac{\mathbf{D}}{m}, \\ \dot{m} &= -\frac{T}{g_0 I_{sp}}, \end{aligned} \tag{78}$$

where $\mathbf{r} = [\, x \;\; y \;\; z \,]^T$ is the Cartesian ECI position, $\mathbf{v} = [\, v_x \;\; v_y \;\; v_z \,]^T$ is the Cartesian ECI velocity, μ is the gravitational parameter, T is the vacuum thrust, m is the mass, g_0 is the acceleration due to gravity at sea level, I_{sp} is the specific impulse of the engine, $\mathbf{u} = [\, u_x \;\; u_y \;\; u_z \,]^T$ is the thrust direction, and $\mathbf{D} = \left[\, D_x \;\; D_y \;\; D_z \,\right]^T$ is the drag force. The drag force is defined in vector form as

$$\mathbf{D} = -\frac{1}{2}\rho S C_D \|\mathbf{v}_{rel}\|\mathbf{v}_{rel}, \tag{79}$$

where C_D is the drag coefficient, S is the reference area, ρ is the atmospheric density, and $\mathbf{v}_{rel}$ is the Earth relative velocity, where $\mathbf{v}_{rel}$ is given as

$$\mathbf{v}_{rel} = \mathbf{v} - \mathbf{\Omega} \times \mathbf{r}, \tag{80}$$

where $\mathbf{\Omega}$ is the angular velocity of the earth relative to the inertial reference frame. The atmospheric density is modeled as the exponential function

$$\rho = \rho_0 \exp[-h/H], \tag{81}$$

where ρ_0 is the atmospheric density at sea level, $h = \|\mathbf{r}\| - R_e$ is the altitude, R_e is the equatorial radius of the earth, and H is the density scale height. The numerical values for these constants can be found in Table II.

15.2.3 *Constraints.* The launch vehicle starts on the ground at rest (relative to the earth) at time t_0, so that the ECI initial conditions are

$$\begin{aligned} \mathbf{r}(t_0) &= \mathbf{r}_0 = \left[\, R_e \cos\phi_0 \;\; 0 \;\; R_e \sin\phi_0 \,\right]^T, \\ \mathbf{v}(t_0) &= \mathbf{v}_0 = \boldsymbol{\Omega} \times \mathbf{r}_0, \\ m(t_0) &= m_0 = 301454 \quad \text{kg}, \end{aligned} \tag{82}$$

where $\phi_0 = 28.5^\circ$ and corresponds to the geocentric latitude of Cape Canaveral, Florida, and it is arbitrarily assumed that the inertially fixed axes are such that the initial inertial longitude is zero. The terminal constraints define the target geosynchronous transfer orbit (GTO), which is defined in terms of orbital elements as

$$\begin{aligned} a_f &= 24361.14 \text{ km}, \\ e_f &= 0.7308, \\ i_f &= 28.5^\circ, \\ \Omega_f &= 269.8^\circ, \\ \omega_f &= 130.5^\circ, \end{aligned} \tag{83}$$

where a is the semimajor axis, e is the eccentricity, i is the inclination, Ω is the inertial longitude of the ascending node, and ω is the argument of perigee. It is noted that, because we are not specifying the location in terminal orbit that must be attained by the vehicle, the true anomaly, ν, is free. These orbital elements can be transformed into ECI coordinates via the transformation, T_{o2c} as given in Bate et al. [1971].

In addition to the boundary constraints, there exists both a state path constraint and a control path constraint in this problem. A state path constraint is imposed to keep the vehicle's altitude above the surface of the earth, so that

$$|\mathbf{r}| \geq R_e, \tag{84}$$

where R_e is the equatorial radius of the earth. Next, the thrust direction is constrained to be of unit length via the equality path constraint

$$\|\mathbf{u}\|^2 = u_x^2 + u_y^2 + u_z^2 = 1. \tag{85}$$

Last, the phases are connected via the following set of linkage constraints at the terminus of phases 1, 2, and 3 and the start of phases 2, 3, and 4, respectively as

$$\begin{aligned} \mathbf{r}(t_f) - \mathbf{r}\big(t_0^{(p+1)}\big) &= \mathbf{0}, \\ \mathbf{v}\big(t_f^{(p)}\big) - \mathbf{v}\big(t_0^{(p+1)}\big) &= \mathbf{0} \qquad (p = 1, \ldots, 3), \\ m\big(t_f^{(p)}\big) - m_{\text{dry}}^{(p)} - m\big(t_0^{(p+1)}\big) &= 0. \end{aligned} \tag{86}$$

It is noted that the linkage constraint on the mass at each of the phase interfaces includes an instantaneous drop of the dry mass of the particular stage. In this case, mass drops at the ends of phases 1, 2, and 3 are given, respectively, as $6m_{\text{dry}}^{\text{srb}}$, $3m_{\text{dry}}^{\text{srb}}$, and $m_{\text{dry}}^{\text{first}}$, where the subscript "dry" denotes the dry mass (i.e., mass excluding fuel) and the superscripts "srb" and "first" denote a single

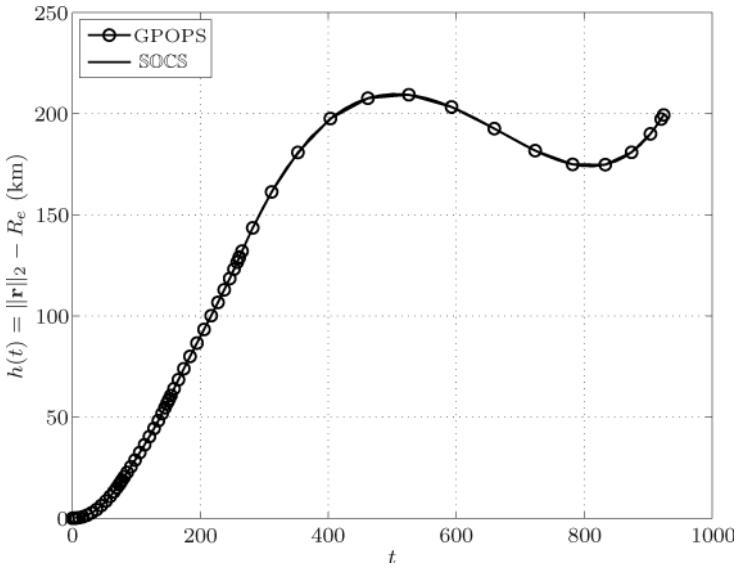

Fig. 9. Altitude versus time for the launch vehicle ascent problem.

solid rocket booster and the first stage, respectively. The objective of the problem is to determine the control (and corresponding trajectory) that minimizes the cost functional

$$J = -m\left(t_f^{(4)}\right) \tag{87}$$

subject to the conditions of Equations (78), (82), (83), (84), and (85). The problem was posed in SI units using the aforementioned auto-scaling procedure and 15 Legendre-Gauss points (i.e., nodes) in each phase. Finally, a constant initial guess was used for the position and velocity in each phase while a linear initial guess was used for the mass.

The GPOPS solution obtained for the launch vehicle ascent problem is summarized in the following Figures 9–12 that contain the altitude, speed, components of control, and Hamiltonian, respectively, alongside the solution obtained using the commercial software program *Sparse Optimal Control Software* (SOCS) [Betts and Huffman 1997]. It is seen that the GPOPS solution and the SOCS solution are in excellent agreement. With regard to accuracy, the optimal objective functions obtained using GPOPS and SOCS are 7529.7123 kg and 7529.7125 kg, respectively, (i.e., resulting in a difference of approximately 2.1×10^{-4} kg). In addition to the excellent agreement of between GPOPS and SOCS, it is seen that the Hamiltonian obtained from GPOPS is piecewise constant. In this case it is known that the durations of the first three phases are *fixed* while the duration of the fourth phase is *free*. Because the Hamiltonian is not an explicit function of time for this problem, we know that its value in the first three phases should be constant (but not necessarily zero) while its value in the fourth phase should be zero (which, indeed, is the case). The additional result showing that the Hamiltonian is constant in each phase (and zero in

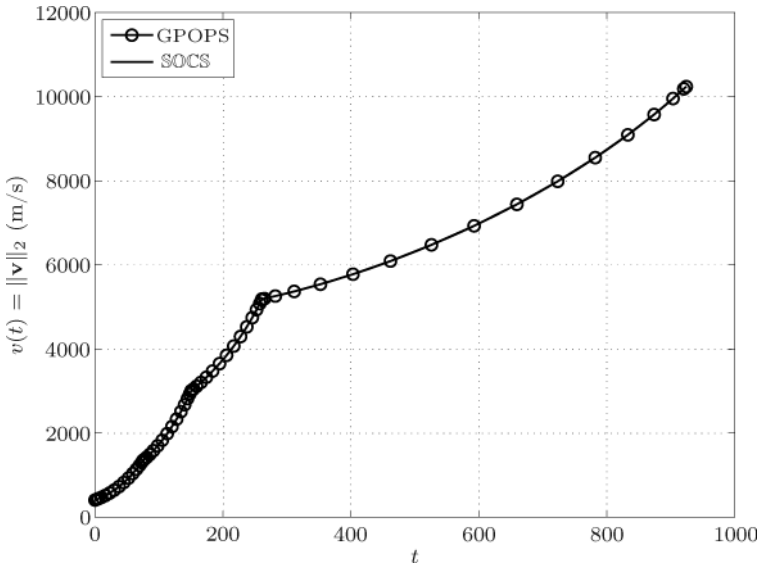

Fig. 10. Inertial speed versus time for the launch vehicle ascent problem.

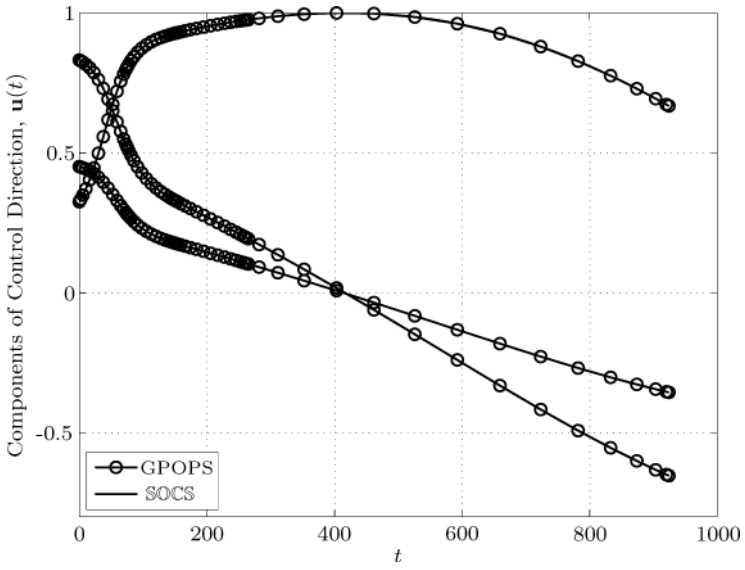

Fig. 11. Components of control versus time for the launch vehicle ascent problem.

the fourth phase) provides further validation of the accuracy of the GPOPS solution.

15.3 Example 3: Goddard Rocket Maximum Ascent Problem

Consider the following well-known optimal control problem [Bryson and Ho 1975] known as the *Goddard rocket maximum ascent problem*. Maximize the

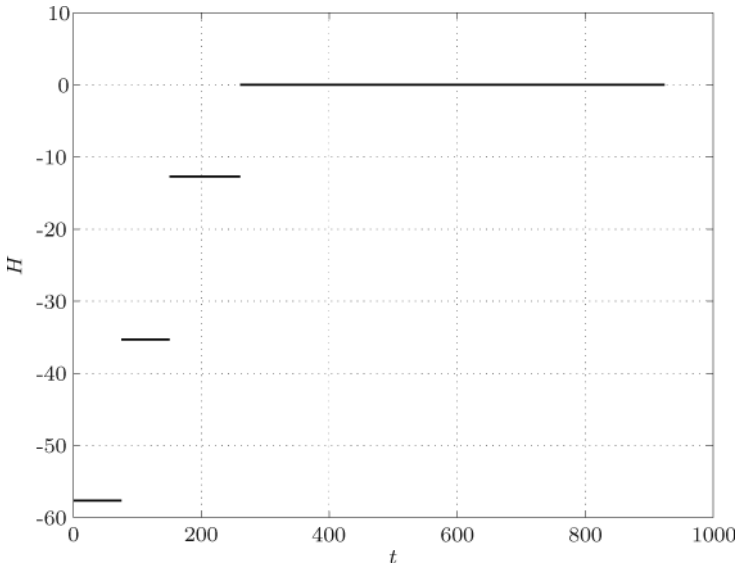

Fig. 12. Hamiltonian versus time for the launch vehicle ascent problem (GPOPS solution only).

cost functional

$$J = h(t_f) \tag{88}$$

subject to the dynamic constraints

$$\begin{aligned} \dot{h} &= v, \\ \dot{v} &= \frac{T - D}{m} - g, \\ \dot{m} &= -\frac{T}{V_e}, \end{aligned} \tag{89}$$

with the boundary conditions

$$\begin{aligned} h(0) &= 0, \\ v(0) &= 0, \\ m(0) &= 3, \\ m(t_f) &= 1, \end{aligned} \tag{90}$$

where h is the altitude, v is the velocity, m is the mass, T is the thrust (and is the control), D is the drag, g is the acceleration due to gravity, and V_e is the exhaust velocity. The drag is modeled as

$$D = D_0 v^2 \exp(-h/H), \tag{91}$$

where $D_0 = 5.49153485 \times 10^{-5}$ $H = 23800$ is the density scale height. For this problem the thrust is constrained so that

$$0 \leq T \leq T_{\max}, \tag{92}$$

where $T_{\max} = 193$.

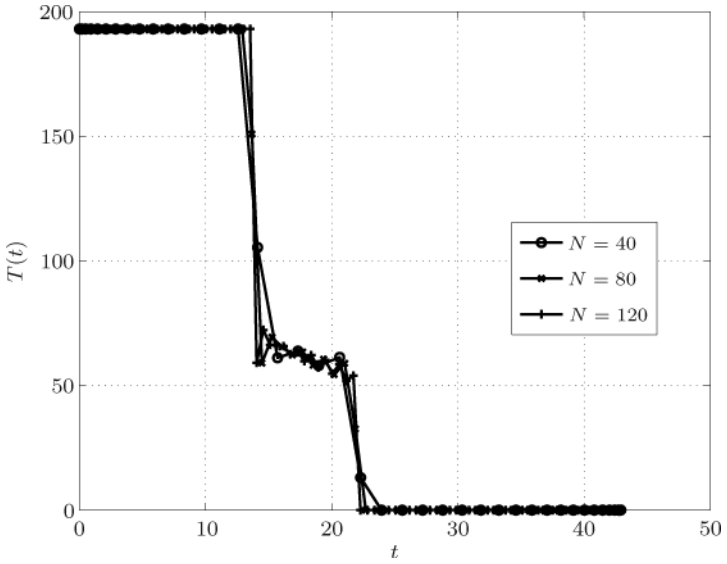

Fig. 13. Thrust versus time for the Goddard rocket maximum ascent problem formulated in GPOPS as a single-phase problem using 40, 80, and 120 LG points.

One of the interesting characteristics of the optimal thrust profile for the Goddard rocket problem is that it contains *singular arc* [Betts 2001], that is, there exists a segment of the solution where the control cannot be determined from the optimality conditions because $\mathcal{H}_{\mathbf{u}}$ and $\mathcal{H}_{\mathbf{uu}}$ are *both* zero during the singular arc (recalling that $\mathcal{H}$ is the Hamiltonian). Commensurate with the fact that the continuous-time optimality conditions are indeterminate during a singular arc, the NLP obtained from a direct collocation method will also have an indeterminacy and the NLP solver will have difficulty determining the optimal control during the singular arc. In this case, the Goddard rocket problem was solved as a one-phase problem using GPOPS without any knowledge of the fact that a singular arc exists in the solution. Figure 13 shows the thrust as a function of time for $N = (40, 80, 120)$. Examining the middle portion of the thrust (where the thrust takes on neither its maximum nor minimum allowable value), it is seen that, while it is no means perfect, the one-phase GPOPS formulation captures the singular arc "reasonably well" as the number of LG points increases. Thus, without any knowledge of the singular arc, the one-phase solution would lead a user to the conclusion that such a behavior exists and could prompt the user to reformulate the problem using multiple phases to obtain a smoother solution during the singular arc.

It is well known that, in order to obtain a more accurate control during the singular arc, it is necessary to impose the singular arc condition which is given as

$$\frac{d}{dt}\left(\mathcal{H}_{\mathbf{uu}}\right) = 0, \tag{93}$$

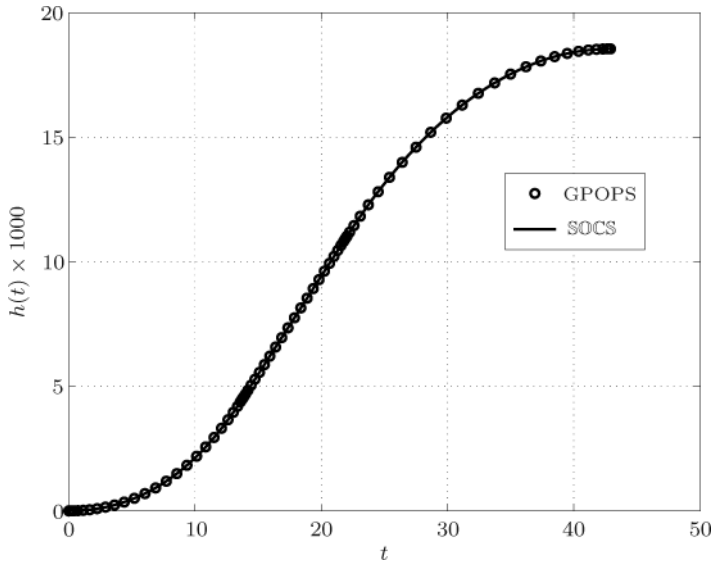

Fig. 14. Altitude versus time for the Goddard rocket maximum ascent problem for 25 LG points using GPOPS alongside SOCS solution.

where $\mathcal{H}$ is the Hamiltonian of the continuous-time optimal control problem and $\mathcal{H}_{\mathbf{uu}}$ is the second derivative of $\mathcal{H}$ with respect to the control. Applying Equation (93), the Goddard rocket problem can be cast as a three-phase optimal control problem with the following conditions added to the original problem [Betts 2001]:

(i) An equality path constraint in phase 2 that arises from Equation (93):

$$\frac{c^2(1+v/c)}{gH-1-2c/v}+\frac{mg}{1+4c/v+2c^2/v^2}=0, \tag{94}$$

where $c = 1580.942528$;

(ii) An event constraint at the end of phase two that places a boundary condition on the end of the singular arc [Bryson and Ho 1975]:

$$mg-(1+v/c)D=0. \tag{95}$$

The three-phase optimal control problem then has the objective of maximizing

$$J=h\left(t_f^{(3)}\right), \tag{96}$$

where the altitude is maximized at the end of phase three. It is noted that the three-phase formulation with the conditions of Equations (94) and (95) is the same as that found in Betts [2001] solved with the software SOCS [Betts and Huffman 1997].

Using the three-phase formulation described above, the Goddard rocket problem was solved with GPOPS using 5 LG (node) points in each phase. This five-node solution was then used as an initial guess to solve the problem with 25 nodes in each phase. The 25-node solutions for $h(t)$, $v(t)$, $m(t)$, and $T(t)$ are shown in Figures 14–17 alongside a solution generated with the software

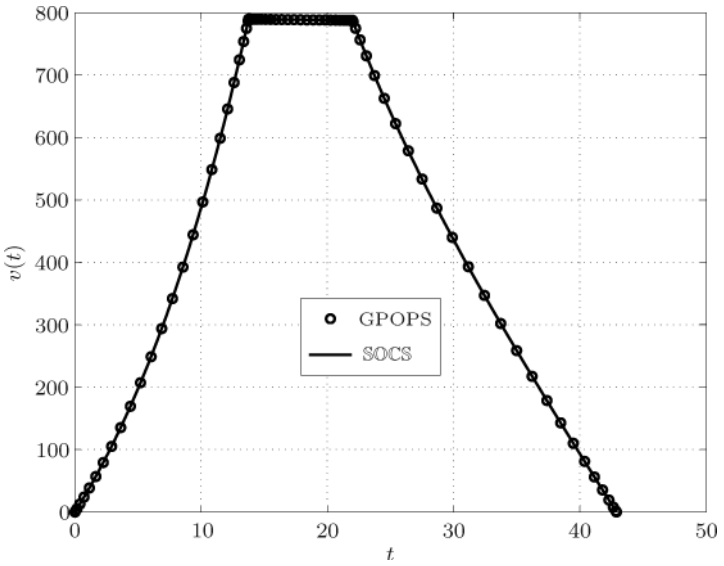

Fig. 15. Velocity versus time for the Goddard rocket maximum ascent problem for 25 LG points using GPOPS alongside SOCS solution.

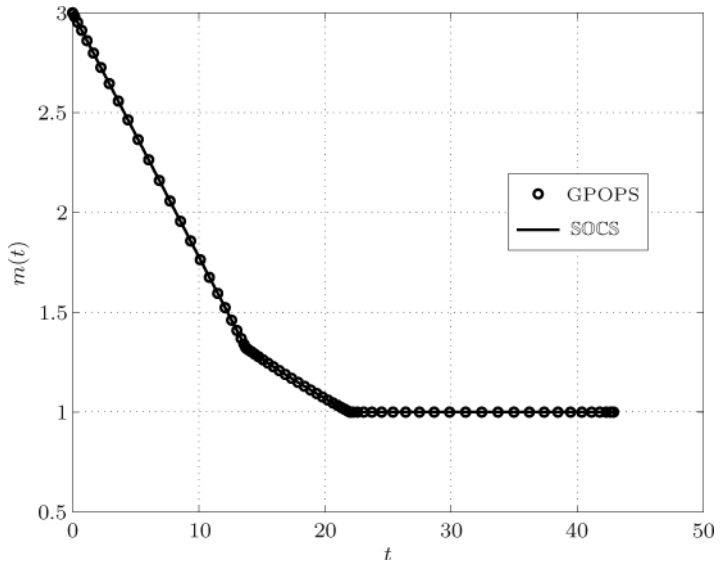

Fig. 16. Mass versus time for the Goddard rocket maximum ascent problem for 25 LG points using GPOPS alongside SOCS solution.

SOCS [Betts and Huffman 1997]. It is seen that, despite the fact that a relatively small number of nodes is used in each phase, the GPOPS solution and the SOCS solution are in excellent agreement. Finally, it is useful to note that the execution time for solving this problem extremely small (on the order of a few seconds) using the built-in automatic differentiator on an Intel Core Duo 2.5-Ghz MacBook Pro running MATLAB R2008b.

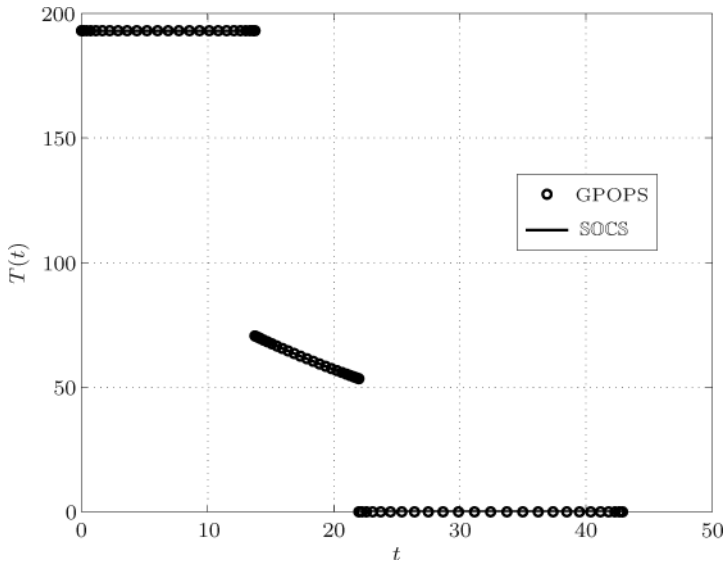

Fig. 17. Thrust versus time for the Goddard rocket maximum ascent problem for 25 LG points using GPOPS alongside SOCS solution.

16. POSSIBLE LIMITATIONS IN THE GPOPS ALGORITHM

It is important to note that, like all software, GPOPS has limitations. First, in the implementation described in this article it is assumed that the state, control, and costate are smooth within any phase of the problem. While for some nonsmooth problems it may be possible to obtain good solutions without increasing the number of phases, it is nevertheless recommended for such problems to ensure smoothness within each phase. Also, while inequality path constraints will be satisfied at the collocation points (i.e., the LG points), in between the collocation points the constraints may be violated (via interpolation using the appropriate basis of Lagrange polynomials). Correspondingly, the costate approximation given in Section 7 may need to be modified for the case of active inequality constraints. Another point related to inequality constraints is that *high-index* path constraints (i.e., path constraints with index greater than 2) can lead to violations in constraint qualifications for fine grids. In such cases, unique Lagrange multipliers do not exist and it is possible that these Lagrange multipliers may become unbounded. Finally, as seen with Example 3 (the Goddard rocket problem), while relatively crude estimates of the control can be obtained during a singular arc, in order to obtain an accurate singular arc control it is necessary to apply the proper singular arc conditions.

17. CONCLUSIONS

An algorithm has been described for solving multiple-phase optimal control problems using the Gauss pseudospectral method (GPM). In particular, a detailed explanation has been given of the mathematical structure of the nonlinear programming problem (NLP) that arises from the multiple-phase GPM.

Many of the implementation details are subsequently discussed. The approach allows the user to design each phase of the optimal control problem independently and then connect any two phases via general linkage conditions on the state and time. In order to alleviate the burden of manually scaling the optimal control problem, an automatic scaling routine has been developed as part of the algorithm. Significant features of the algorithm include the separation of the constant derivatives of the constraint Jacobian from the nonconstant derivatives, highly accurate computation of the costates (adjoints), and the use of automatic differentiation. A reusable MATLAB software implementation, called *GPOPS*, has been included with the algorithm developed and has been demonstrated successfully on three examples.

AVAILABILITY OF SOFTWARE

The GPOPS software can be downloaded in the form of MATLAB source code at no charge from online.[11] Additional infomration about GPOPS can also be obtained online.[12]

ACKNOWLEDGMENTS

The authors acknowledge Dr. John Betts of The Boeing Company for providing solutions obtained using the software SOCS for comparison with GPOPS and for his time in discussing many important issues in the numerical solution of optimal control problems. The authors also gratefully acknowledge United States Air Force Lt. Col. (Dr.) Timothy Jorris for his immense help in both debugging the GPOPS software and for his numerous helpful suggestions in developing the interface for the software. In addition, we thank Dr. Jorris for providing us with MATLAB code that helped us develop both the GPOPS library and the built-in automatic differentiator.

REFERENCES

Balsa-Canto, E., Banga, J. R., Alonso, A. A., and Vassiliadis, V. S. 2001. Dynamic optimization of chemical and biochemical processes using restricted second-order information. *Comput. Chem. Eng. 25*, 4–6, 539–546.

Bate, R. R., Mueller, D. D., and White, J. E. 1971. *Fundamentals of Astrodynamics*. Dover Publications, New York, NY.

Becerra, V. M. 2009. *PSOPT Optimal Control Solver User Manual*. University of Reading, Reading, U.K. http://www.psopt.org.

Benson, D. A. 2004. A gauss pseudospectral transcription for optimal control. Ph.D. dissertation. MIT, Cambridge, MA.

Benson, D. A., Huntington, G. T., Thorvaldsen, T. P., and Rao, A. V. 2006. Direct trajectory optimization and costate estimation via an orthogonal collocation method. *J. Guid. Contr. Dynam. 29*, 6, 1435–1440.

Betts, J. T. 1998. Survey of numerical methods for trajectory optimization. *J. Guid. Contr. Dynam. 21*, 3, 193–207.

Betts, J. T. 2001. *Practical Methods for Optimal Control Using Nonlinear Programming*. SIAM, Philadelphia, PA.

[11] http://www.sourceforge.net/projects/gpops.

[12] http://gpops.sourceforge.net.

Betts, J. T. and Frank, P. D. 1994. A sparse nonlinear optimization algorithm. *J. Optimiz. Theor. Appl. 82*, 2, 193–207.

Betts, J. T. and Huffman, W. P. 1997. *Sparse Optimal Control Software—SOCS*, MEA-LR-085 Ed. Boeing Information and Support Services, Seattle, WA.

Bryson, A. E. and Ho, Y.-C. 1975. *Applied Optimal Control*. Hemisphere Publishing, New York, NY.

Byrd, R. H., Nocedal, J., and Waltz, R. 2006. *KNITRO: An Integrated Package for Nonlinear Optimization*. University of Colorado Boulder, CO, and Northwestern University, Evanston, IL.

Canuto, C., Hussaini, M. Y., Quarteroni, A., and Zang, T. A. 1988. *Spectral Methods in Fluid Dynamics*. Spinger-Verlag, Heidelberg, Germany.

Canuto, C. G., Hussaini, M. Y., Quarteroni, A., and Zang, T. A. 2007. *Spectral Methods: Evolution to Complex Geometries and Applications to Fluid Dynamics*. Spinger-Verlag, Heidelberg, Germany.

Cizniar, M., Fikar, M., and Latifi, M. A. 2006. Matlab dynamic optimization code dynopt user's guide. `http://www.kirp.chtf.stuba.sk/fikar/research/dynopt/dynopt.htm`.

Curtis, A. R., Powell, M. J. D., and Reid, J. K. 1974. On the estimation of sparse Jacobian matrices. *J. Inst. Math. Appl. 13*, 1, 117–120.

Davis, P. and Rabinowitz, P. 1984. *Methods of Numerical Integration*. Academic Press, New York, NY.

Don, W.-S. 2000. Pseudopack: A software for solving computational fluid dynamics problems. Tech. rep. Brown University, Providence, RI.

Elnagar, G. and Kazemi, M. 1998. Pseudospectral chebyshev optimal control of constrained nonlinear dynamical systems. *Computat. Optimiz. Appl. 11*, 2, 195–217.

Elnagar, G., Kazemi, M., and Razzaghi, M. 1995. The pseudospectral legendre method for discretizing optimal control problems. *IEEE Trans. Aut. Contr. 40*, 10, 1793–1796.

Fahroo, F. and Ross, I. M. 2000. A spectral patching method for direct trajectory optimization. *J. Astronaut. Sci. 48*, 2–3, 269–286.

Fahroo, F. and Ross, I. M. 2001. Costate estimation by a legendre pseudospectral method. *J. Guid. Contr. Dynam. 24*, 2, 270–277.

Fornberg, B. 1994. A review of pseudospectral methods for solving partial differential equations. *Acta Numerica*, 203–267.

Fornberg, B. 1998. *A Practical Guide to Pseudospectral Methods*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, Cambridge, U.K. New York, NY.

Forth, S. A. 2006. An efficient overloaded implementation of forward mode automatic differentiation in matlab. *ACM Trans. Math. Softw. 32*, 2, 195–222.

Forth, S. A. and Edvall, M. M. 2007. *User's Guide for MAD—A MATLAB Automatic Differentation Toolbox (TOMLAB/MAD) Version 1.4*. TOMLAB Optimization, Inc., Pullman, WA.

Gill, P. E., Murray, W., and Saunders, M. A. 2002. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM J. Optimiz. 12*, 4, 979–1006.

Gill, P. E., Murray, W., and Saunders, M. A. 2006. *User's Guide for SNOPT Version 7: Software for Large Scale Nonlinear Programming*. Stanford University, Palo Alto, CA, and University of California, San Diego, La Jolla, CA.

Goh, C. J. and Teo, K. L. 1988. Miser: A Fortran program for solving optimal control problems. *Adv. Eng. Softw. 10*, 2, 90–99.

Huntington, G. and Rao, A. V. 2007. A comparison of accuracy and computational efficiency of three pseudospectral methods. In *Proceedings of Guidance, Navigation and Control Conference*. American Institute of Aeronautics and Astrodynamics, Washington, DC.

Huntington, G. T. 2007. Advancement and analysis of a gauss pseudospectral transcription for optimal control. Ph.D. dissertation. Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA.

Huntington, G. T., Benson, D. A., and Rao, A. V. 2007. Design of optimal tetrahedral spacecraft formations. *J. Astronaut. Sci. 55*, 2, 141–169.

Huntington, G. T. and Rao, A. V. 2008a. A comparison of global and local collocation methods for optimal control. *J. Guid. Contr. Dynam. 31*, 2, 432–436.

HUNTINGTON, G. T. AND RAO, A. V. 2008b. Optimal reconfiguration of spacecraft formations using the gauss pseudospecral method. *J. Guid. Contr. Dynam. 31*, 3, 689–698.

JANSCH, C., WELL, K. H., AND SCHNEPPER, K. 1994. GESOP—eine software umgebung zur simulation und optimierung. *Proceedings des SFB*.

JOCKENHOVEL, T. 2002. Optcontrolcentre, software package for dynamic optimization. `http://OptControlCentre.com/`.

KAMESWARAN, S. AND BIEGLER, L. T. 2008. Convergence rates for direct transcription of optimal control problems using collocation at Radau points. *Comput. Optimiz. Appl. 41*, 1(Sep.), 81–126.

KIRK, D. E. 2004. *Optimal Control Theory: An Introduction*. Dover Publications, New York, NY.

MARTINS, J. R. R., STURDZA, P., AND ALONSO, J. J. 2003. The complex-step derivative approximation. *ACM Trans. Math. Softw. 29*, 3, 245–262.

RAO, A. V. 2003. Extension of a pseudospectral legendre method for solving non-sequential multiple-phase optimal control problems. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, (Austin, TX). AIAA paper 2003-5634.

ROSS, I. M. AND FAHROO, F. 2001. *User's Manual for DIDO 2001 α: A MATLAB Application for Solving Optimal Control Problems*. Tech. rep. AAS-01-03. Department of Aeronautics and Astronautics, Naval Postgraduate School, Monterey, CA.

ROSS, I. M. AND FAHROO, F. 2004a. Legendre pseudospectral approximations of optimal control problems. In *New Trends in Nonlinear Dynamics and Control, and Their Applications*, W. Kang, M. Xiao, and C. Borges, Eds. Lecture Notes in Control and Information Sciences, Vol. 295. Springer-Verlag, Heidelberg, Germany, 327–342.

ROSS, I. M. AND FAHROO, F. 2004b. Pseudospectral knotting methods for solving optimal control problems. *J. Guid. Contr. Dynam. 27*, 3, 397–405.

ROSS, I. M. AND FAHROO, F. 2008a. Advances in pseusospectral methods. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference* (Honolulu, HI). AIAA paper 2008-7309.

ROSS, I. M. AND FAHROO, F. 2008b. Convergence of the costates do not imply convergence of the controls. *J. Guid. Contr. Dynam. 31*, 4, 1492–1497.

RUMP, S. 2008. Intlab - interval laboratory. `http://www.ti3.tu-harburg.de/rump/intlab/`.

RUTQUIST, P. AND EDVALL, M. 2008. *PROPT: MATLAB Optimal Control Software*. Tomlab Optimization, Inc. Pullman, WA.

The Mathworks, Inc. 2008. *MATLAB: A Language for Technical Computing*. The Mathworks, Inc., Natick, MA.

TREFETHEN, L. N. 2001. *Spectral Methods in MATLAB*. SIAM Press, Philadelphia, PA.

VLASES, W. G., PARIS, S. W., MARTENS, M. J., AND HARGRAVES, C. R. 1990. Optimal trajectories by implicit simulation. Tech. rep. WRDC-TR-90-3056. Boeing Aerospace and Electronics, Seattle, WA.

VLASSENBROECK, J. AND DOREEN, R. V. 1988. A Chebyshev technique for solving nonlinear optimal control problems. *IEEE Trans. Automat. Contr. 33*, 4, 333–340.

VON STRYK, O. 2000. *User's Guide for DIRCOL (Version 2.1): A Direct Collocation Method for the Numerical Solution of Optimal Control Problems*. Technical University, Munich, Germany.

WEIDEMAN, J. A. C. AND REDDY, S. C. 2000. A MATLAB differentiation matrix suite. *ACM Trans. Math. Softw. 26*, 4, 465–519.

WILLIAMS, P. 2004a. Application of pseudospectral methods for receding horizon control. *J. Guid. Contr. Dynam. 27*, 2, 310–314.

WILLIAMS, P. 2004b. Jacobi pseudospectral method for solving optimal control problems. *J. Guid. Contr. Dynam. 27*, 2, 293–297.

WILLIAMS, P. 2005. Hermite-Legendre-Gauss-Lobatto direct transcription methods in trajectory optimization. (AAS 05-131). *Adv. Astronaut. Soci. 120*, part 1, 465–484.

WILLIAMS, P. 2008. *User's Guide for DIRECT 2.0*. Royal Melbourne Institute of Technology, Melbourne, Australia.

Received October 2008; revised February 2009, June 2009; accepted June 2009